An Implementation of the
MPEG-2 Audio Decoding Specification

by

Chad Mikkelson

Submitted to the department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology
May 1995

Author______ ________________

Department of Electrical Engineering and Computer Science
May 26, 1995

Certified by_____ ________________________

Panos Papamichalis
Thesis Supervisor

Certified by____

V. Michael Bove
Thesis Supervisor

Accepted by______ __________________

F. R. Morgenthaler
Chairman, Department Committee on Graduate Theses

An Implementation of the
MPEG-2 Audio Decoding Specification
by
Chad H. Mikkelson

Submitted to the
Department of Electrical Engineering and Computer Science

May 17, 1995

In Partial Fulfillment of the Requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The MPEG committee has developed standards for the compression of digital audio and video information. The two audio coding specifications are: MPEG-1 for compressing mono or stereo audio signals and MPEG-2 audio for compressing five channel stereo audio signals. This thesis describes a study of the MPEG audio system and an implementation of the MPEG-2 Audio standard in the C programming language and on a Texas Instruments TMS320C30 digital signal processing chip. After completion of the implementation, both the C version and the DSP version of the decoder were tested for accuracy, backwards compatibility, and performance. The test results and a performance analysis are included in this document.

M.I.T. Thesis Supervisor: V. Michael Bove
Title: Associate Professor, Media Arts and Sciences

Texas Instruments Thesis Supervisor: Panos Papamichalis
Title: Branch Manager, DSP Applications

Table Of Contents

List Of Figures

List Of Tables

1. Introduction

The Motion Picture Experts Group, MPEG, is a committee of the International Standards Organization, ISO, with the purpose of standardizing compression for digital video and associated audio signals. The MPEG committee has two standards for compression of digital audio: the standard officially known as ISO/IEC 11172-3[1] (MPEG-1) for two channel stereo signals and a second known as ISO/IEC 13818-3[2] (MPEG-2) for five channel stereo signals. The MPEG-1 standard was adopted in 1992 and since then software packages and integrated circuits have been developed which compress and decompress digital audio. These MPEG-1 encoders and decoders have since been incorporated into consumer products. The MPEG-2 standard was approved in November, 1994 and its potential for application and development for the consumer market is now being evaluated.

This thesis describes a study of the MPEG audio system and an implementation of the MPEG-2 Audio standard in the C programming language and on a Texas Instruments TMS320C30 digital signal processing chip. Following completion of the implementation, both the C version and the DSP version of the decoder were tested for accuracy, backwards compatibility, and performance. Both of the decoder implementations decoded the MPEG bit streams correctly. The test results and analysis are included in this document.

1.1 Algorithm Overview

An uncompressed digital audio signal at a 48 kHz sample rate, with 16 bit samples, and two channels, requires 1.5 Mbit/s to transmit or store the information. A full five channel system would require two and a half times as many bits per second to be transmitted or stored. Since audio/video multimedia must allocate the bulk of their bandwidth to the video portion of the program, the bandwidth available for audio is fairly limited. With an uncompressed audio signal, it would generally not be possible to include three extra channels without reducing the quality of the video signal. The objective of the MPEG audio system is to reduce the amount of data required for a given quality of audio to be perceived by a listener.

Audio data is compressed by two different methods: reduction of redundancy in the source, and removal of irrelevant data with respect to the sink (the listener). Redundancy in the source exists when there is a non-uniform probability mass function. Removing redundancy results n a uniform probability mass function and a maximum amount of information transmitted per bit. A model of the sink is used to determine which parts of the signal are irrelevant. With regard to a human listener, the irrelevant data might be a sine wave of an amplitude too low to hear, a frequency outside the range of hearing, or quantization noise.[3]

The MPEG audio coding algorithm uses both removal of redundant and of irrelevant data. Since the algorithm actually removes data, it is a lossy compression technique. The MPEG algorithm uses a technique known as subband filtering to code an audio signal. That is, the input audio signal is split into equal-width frequency subbands. Then a variable number of bits is allocated to each subband quantizer with the goal of minimizing the perceivable quantization error[4]. To determine the relative effect of quantizing the data, a model of the human auditory system called the psychoacoustic model is used. The signal is compressed (coded with fewer bits) if the number of bits used to quantize the samples is less than the number of bits used to code the samples originally.

MPEG is a syntax specification, meaning the coded bit stream transmitted from the encoder to the decoder is rigidly specified, and much of the decoding process is also specified. The encoder may use various means of determining how to code the input data but the output of the encoder must conform to the specification. In addition, the decoding and interpretation of the MPEG data is specified. The sample reconstruction operations and the subband synthesis operations are also specified, but not their implementation.

The basic structure of the system is shown in Figure 1. A complete MPEG audio system consists of an encoder and a decoder. The purpose of the encoder is to compress the input PCM audio stream and to format the data for transmission to the decoder. To facilitate compression, the samples of each channel are divided by time into frames of approximately 8 ms or 24 ms in length. A filter bank is used to separate the input signal into subsampled subbands where each subband contains some fraction of the full input frequency spectrum. The psychoacoustic model is used to determine the number of bits allocated to the samples in the different subbands. The optimal bit allocation minimizes the noise introduced by quantizing the samples audible to the listener. The encoded subband samples, the bit allocation information, and other side information is packed into a frame and either stored or transmitted to the decoder.

The responsibility of the MPEG decoder is to take the MPEG encoded input stream, decode it, and convert it to a PCM audio stream. The decoding process can be broken into several parts: unpacking the bit stream, restoring the samples by dequantizing and rescaling, and performing the synthesis filtering operation. Unpacking the bit stream is complicated because the exact format of the frame is not known until the header has been unpacked and analyzed. In addition, the number of bits in the samples is known only after the bit allocation has been decoded. When all of the sample data has been unpacked, the samples are dequantized according to the bit allocation and converted back to a full bandwidth time domain signal by the synthesis filter.

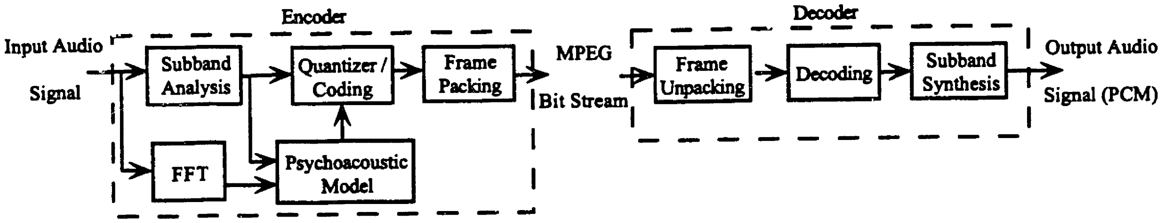

Figure 1 System Encoder/Decoder

1.2 MPEG-1 Overview

There are three different coding layers in the MPEG system, each one more complex but also more efficient. Layers I and II are commonly supported in coding systems. Layer III produces a higher coding gain but requires more processing time. The MPEG frame, in layer I, consists of the data for 384 samples per channel while a layer II frame has three times as many or 1152 samples per channel. Since the layer II frames are three times as long, the overhead information is a smaller fraction of the frame. Layer II also introduces further compression with respect to Layer I by redundancy and irrelevancy removal on the scalefactors and uses a more compact representation of the quantized values. Samples may also be grouped together in Layer II for further compression. Layer III frames use a hybrid filter bank to increase frequency resolution in the subbands, a non-uniform quantizer, and Huffman coding of the quantized samples. When a layer II frame of 1152 samples is subband filtered into the 32 subbands every subband will contain 36 samples. Layer II frame are sometimes described in terms of subframes, where each subframe contains 384 samples or 12 samples in each subband. Likewise each subband in layer I would contain 12 samples. The input data is also converted to the frequency domain by a Fast Fourier Transform. The conversion to the frequency domain is necessary to apply the psychoacoustic model which determines the ear's sensitivity to noise in each subband. The level below which the ear can not hear noise is called the masking threshold. Bits are allocated to the samples in each subband with the aim of keeping the quantization noise below the masking threshold and producing the best subjective sound quality with the available bandwidth. The subband samples in a frame are normalized by dividing the samples by a scalefactor and the result of the division is quantized by the number of bits in the bit allocation. A subband sample is now represented by three numbers: the scalefactor, the number of bits in the bit allocation, and the quantized subband sample with the same scalefactor and bit allocation used for all the samples in each subband of a frame.

The filter banks in an MPEG audio coder (or any subband coder) are critically important. They are known as perfect reconstruction filter banks because of their flat frequency response and aliasing cancellation properties. These flat frequency response is important to maintain the maximum dynamic range of the samples within each subband and an equal weight on all frequencies. The cancellation property is also necessary since aliasing between the subbands would cause audible artifacts.

The number of bits available for allocating to the samples depends on the bit rate, the sample rate, and the type of frame. The bit rate for MPEG-1 may vary up to 384 Kbits/sec; there are three available sample rates: 32, 44.1, and 48 kHz. As the bit rate decreases or the sample rate increases, the number of bits available per sample decreases.

Another property of MPEG audio frames is that they may contain a Cyclic Redundancy Check word for performing error detection. The CRC word is calculated using the bits that are most critical to correctly decoding the frame correctly: the header, the bit allocation, and the scalefactors. The CRC word is only used to do error detection, so if an error is detected in a frame, the output of the frame is muted or the previous frame repeated.

If a stereo bit stream is being encoded, a method called joint-stereo encoding may be used to further increase compression. The method exploits similarities between the left and right signals and the ear's inability to localize sounds at high frequencies (above 2 kHz) by using the same bit allocation and sample data for each channel, but different scalefactors.

1.3 MPEG-2 Overview

MPEG-2 is an extension to MPEG-1 developed for compressing multichannel audio. In addition to the left and right channels, there is an additional center channel in front, and two rear surround channels, left surround and right surround. The format is known as 3/2 stereo and requires the transmission of five appropriately formatted audio signals. The additions to the MPEG-1 encoder and decoder are the ability to process up to six channels of audio, functions that facilitate the backwards compatibility, and functions that reduce the number of bits needed for coding the extra channels.

Backwards compatibility in the MPEG system is defined as having an MPEG-2 encoder that will produce a stream which can be decoded by an MPEG-1 decoder and an MPEG-2 decoder which can decode a stream coded by an MPEG-1 encoder. The MPEG-2 encoder creates a backward compatible stream by using matrixing and by using a frame structure similar to MPEG-1. In the matrixing procedure, shown in Figure 2, the left, left surround, and center channels are added together and the combined signal is transmitted as the left compatible channel. A similar process is carried out to produce a right compatible channel. The channels created by matrixing are called compatible channels because they are placed in the frame the same way as two stereo channels in MPEG-1 and may be decoded with an MPEG-1 decoder. When the backwards compatible stream is decoded by an MPEG-1 decoder, the two output channels, the left and right are a combination of all five original channels. The MPEG-2 frame may contain up to three channels in addition to the left and right compatible channels. These three channels are chosen so that when an MPEG-2 decoder decodes the stream, all five original channels may be recovered by dematrixing.

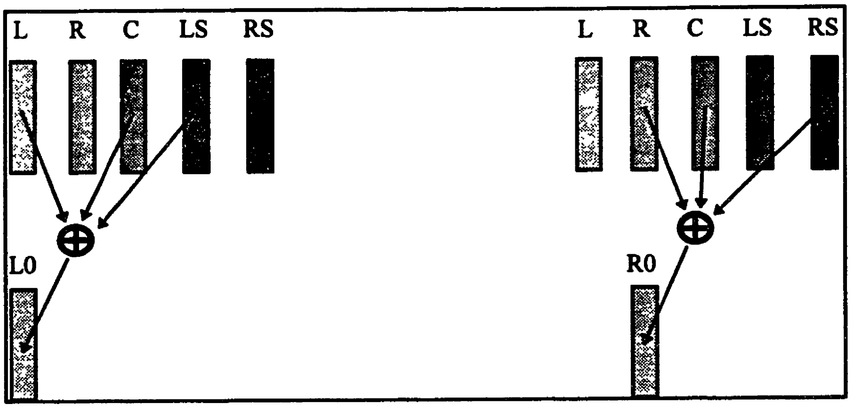

Figure 2 Matrixing Operation

An MPEG-2 audio frame is shown in the figure below. The two MPEG-1 compatible channels are placed in the MPEG-1 audio data section and up to three other channels are placed in the multichannel audio data section. The structure of the first part of the frame, the header, CRC word, and the audio data, is identical to an MPEG-1 frame. The multichannel information at the end of the frame, would be considered ancillary data by an MPEG-1 decoder.

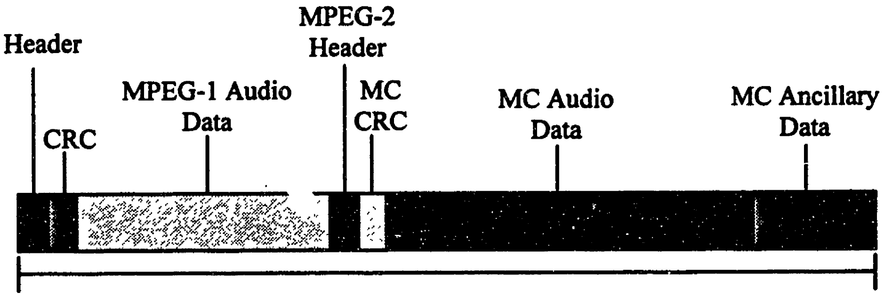

Figure 3 MPEG-2 Audio Frame

Two additional compression methods have been added to MPEG-2 to facilitate compression for the extra channels: Multichannel Prediction and Dynamic Crosstalk. Multichannel Prediction looks for statistical dependencies between the multichannels and the left or right channel. If there is a strong correlation, then only the difference between the two channels is transmitted. While Multichannel Prediction finds statistical dependencies between channels, Dynamic Crosstalk determines which parts of the stereophonic signal are irrelevant with respect to the spatial perception of the presentation (i.e. which parts do not contribute to the localization of sound).[1] At high frequencies (above 2 kHz) the localization of the stereophonic image is determined by the temporal envelope and not by the temporal fine structure of the audio signal. Since the temporal envelope is determined by the scalefactors and the fine structure by the subband samples, a combination of samples transmitted in one channel may be multiplied by scalefactors in the same channel and the scalefactors in a different channel, resulting in two signals with the same fine structure but a different temporal envelope. Using Dynamic Crosstalk saves transmitting the bit allocations and subband samples in one channel while preserving the spatial perception of the sound.

The MPEG-2 standard also provides a way to increase the bit rate of the system to improve the quality of the audio with an extension bit stream. Without the extension bit stream, five channels of audio would have to use the number of bits previously used for only two channels. For every MPEG-1 compatible audio frame, a frame from the extension stream is also available. The Extension stream provides the capability to increase the bit rate to 896 Kbits/sec.

1.4 Applications

The MPEG audio system is designed for systems that store or transmit audio in digital formats. These systems include consumer and professional applications such as digital audio broadcasting, direct digital television transmission, recording, and multimedia. For instance, the Direct Digital Satellite television system uses MPEG-1 coding of both the video and the audio signal. A complete MPEG system (audio and video) could be included in a camcorder to increase the length of a movie that may be recorded or they could be included with a PC to record and transmit video clips. The current applications for the MPEG-2 system are mainly in home theater, where an MPEG-2 decoder would produce a full five channel stereo output that could be played back on a surround sound system. As multichannel coding technology improves, the number of applications will only grow.

2. MPEG Algorithm

This section describes the different parts of the MPEG-1 and MPEG-2 algorithms including: subband filtering, the psychoacoustic model, the representation of samples, CRC error checking, matrixing of channels, transmission channel allocation, dynamic crosstalk, and multichannel prediction. The first three sections on subband filtering, the psychoacoustic model, and the representation of samples are coding blocks that are present in both MPEG-1 and MPEG-2 encoders. CRC error checking is optional but can be included in both encoders. Finally matrixing, dynamic crosstalk, and multichannel prediction are present only in MPEG-2 encoders and are used to achieve compatibility with MPEG-1 and to implement compression of multichannel audio signals.

2.1 Subband Filtering

The MPEG algorithm uses a technique known as subband filtering to code an audio signal. In the encoder, the input audio signal is split into equal-width frequency subbands by a subband analysis filter. Then a variable number of bits is allocated to each subband and the samples in that subband are quantized according to the bit allocation. Since the samples are coded with fewer bits than the original signal, the signal may be transmitted with a lower bit rate or stored in a smaller amount of memory. In the decoder the samples are first requantized and then must be recombined into the entire frequency spectrum by the subband synthesis filter. A block diagram of the filtering operation is shown in Figure 4 below.

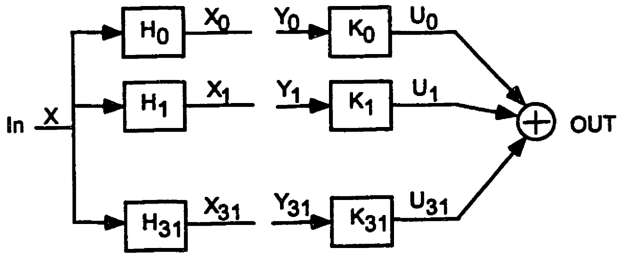

Figure 4 Subband Filtering

Since the samples in each subband occupy only a fraction of the Nyquist bandwidth of the original signal, the signal in each subband may be downsampled. If the signal is downsampled by the number of subbands, then the total number of samples input to the filter bank will equal the total number of samples out (critical sampling). To reconstruct the original signal the subbands are upsampled in the synthesis filter.

The subband filters are created by constructing a prototype filter and translating it to the proper position in frequency. Thus, the subbands in the filter bank are equal width in frequency. The frequency response and the impulse response of the prototype filter is shown in below in Figure 5 below. The prototype is a lowpass filter that has been numerically optimized to satisfy the perfect reconstruction properties. Since it is not possible to implement a filter with infinitely sharp cutoff, there will be some overlap between the adjacent filters. The overlap between the filters will result in aliasing following the decimation operation. It is possible to choose filters so that the aliasing components introduced by the downsampling process cancel when the subbands are recombined[5]. It is also important that the analysis filter bank has a (nearly) flat frequency response for all frequencies, to maintain the full signal range in all the subbands.

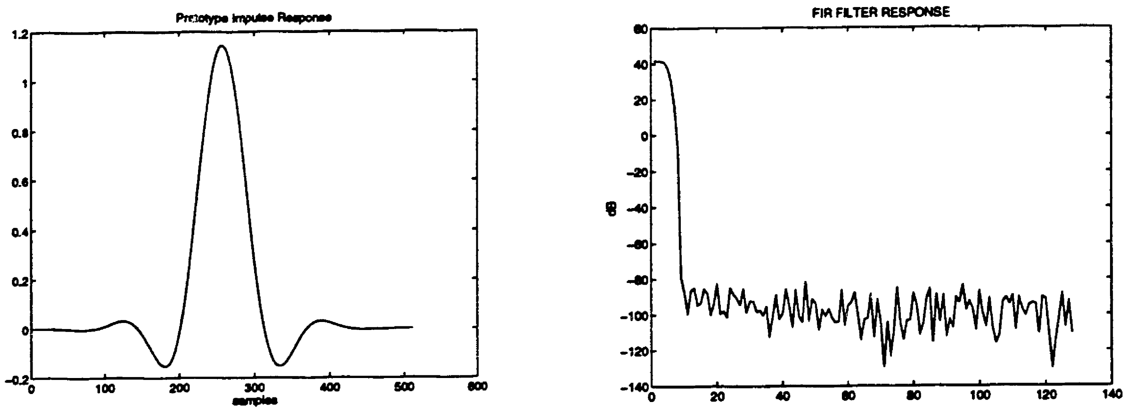

Figure 5 Prototype Filter Response

The number of subbands in a subband filter bank determines the frequency resolution. Since the bandwidth of the input signal is fixed by the sample rate, with more subbands the frequency width of each subband is smaller. The length of the filter (the length of the impulse response in samples) determines the time resolution and aliasing between subbands. MPEG-1 layer 1 and 2 use 32 subband, 512 length filter banks. MPEG-1 layer 3 uses 192 or 384 subband "hybrid" filter banks. Frequency resolution is important in locating tones (and noise) that will affect the masking threshold. To make up for the limited frequency resolution of the filter bank, a Fast Fourier Transform may be used in parallel with the filter bank to determine the spectrum of the input signal.

2.1.1 Quadrature Mirror Implementation

One of the earliest implementations of a perfect reconstruction filter banks was the Quadrature Mirror Filter Bank by Esteban and Galand[4]. The algorithm used two equal width filter banks to split the signal into subbands. A block diagram of the filter is shown below in Figure 6. The filter $H_1(z)$ is a symmetric low pass filter and $H_2(z)$ is a high pass filter. The filters satisfy a mirror relationship so:

$$\left|H_1\left(e^{j\omega T}\right)\right| = \left|H_2\left(e^{j(\frac{\omega_s}{2}-\omega)T}\right)\right| \quad \text{where } \omega_s=2\pi f_s=2\pi/T \text{ is the sample rate.} \qquad (1)$$

The output signals from the filters occupy approximately one-half the original Nyquist bandwidth so the signals may be downsampled by a factor of two. Since the filters have a finite transition band, there will be some aliasing as a result of the downsampling operation. To reconstruct the full bandwidth signal, the subband signals are upsampled by two and then filtered by $K_1(z)$ and $K_2(z)$ to select the proper frequency-scaled images. If the filters are chosen properly, the aliasing components from the adjacent subbands cancel, and the signal may be perfectly reconstructed. The filters may be cascaded together to produce more than two bands.

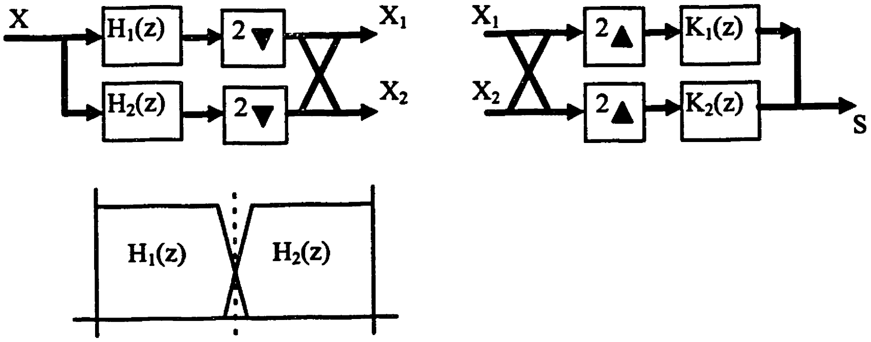

Figure 6 Block Diagram of a Subband Filter

The output from the filter bank is

$$S(z) = \frac{1}{2}\{H_1(z)K_1(z) + H_2(z)K_2(z)\}X(z)$$
$$+\frac{1}{2}\{H_1(-z)K_1(z) + H_2(-z)K_2(z)\}X(-z) \qquad (2)$$

Where the last two terms represent aliasing from the downsampling operation. To meet the perfect reconstruction constraints, we must select the filters so the aliasing terms cancel. To satisfy (1), the filters $H_1(z)$ and $H_2(z)$ must be chosen so

$$H_2(z) = H_1(-z) \qquad (3)$$

then by setting

$$K_1(z) = H_1(z) \text{ and } K_2(z) = -H_2(z) = -H_1(-z) \qquad (4)$$

the aliasing terms will cancel and we will be left with

$$S(z) = \frac{1}{2}\left\{H_1^2(z) - H_1^2(-z)\right\}X(z) \qquad (5)$$

evaluating the result on the unit circle and simplifying we get

$$S(e^{j\omega T}) = \frac{1}{2}\left\{H_1^2(\omega) + H_1^2(\omega + \frac{\omega s}{2})\right\}e^{-j(N-1)\omega T}X(e^{j\omega T}) \qquad (6)$$

if the filters satisfy the condition that the magnitude response of the filter bank is constant

$$H_1^2(\omega) + H_1^2\left((\omega + \frac{\omega s}{2}\right) = 1 \qquad (7)$$

then the output of the filter bank will be a scaled and delayed version of the input

$$S(e^{j\omega T}) = \frac{1}{2}e^{-j(N(-1)\omega T} \bullet X(e^{j\omega T}) \qquad (8)$$

Thus the aliasing between the subbands can be canceled with a proper construction of the subband filters. The filter bank in MPEG uses a multiband generalization of this technique.

2.1.2 Multiband Filters

The Quadrature Mirror Filter principle may be extended to allow the construction of filter banks with any number of equal size filter banks by simply cascading the stages of the filter banks. However, the filter bank may be implemented with lower computational complexity with other techniques such as a polyphase quadrature filter[5] or a Modified Discrete Cosine Transform[6]. The polyphase filter bank implementation is used in MPEG layers I and II. The polyphase implementation in MPEG uses 32 equally spaced filter banks, each created by a translating a prototype filter. A multiband subband filter is shown below in Figure 7. This multiband filter meets similar requirements as the Quadrature Mirror Filter, namely the cancellation of aliasing between adjacent bands and that the filter shapes be designed such that the adjacent filters add to produce a flat frequency response.

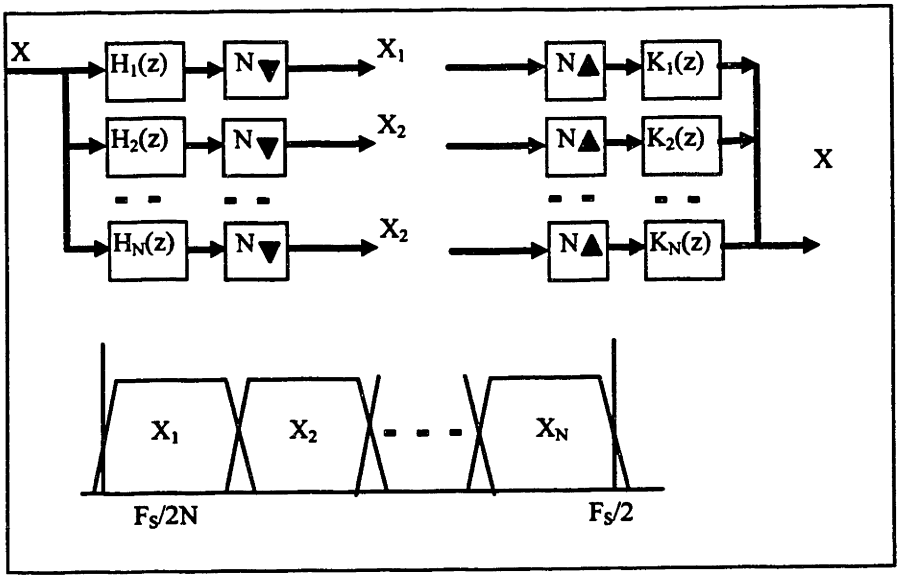

Figure 7 Multiband Filter

2.2 Psychoacoustic Model

Some properties of human auditory perception can be used to reduce irrelevancy in the signal to reduce the bitrate required to maintain the desired quality. Two effects may make part of the input signal irrelevant to the listener. The first is that the ear is more sensitive to sounds in the midrange frequencies (between 1 kHz and 15 kHz) than the high or low frequencies. So one strategy is to identify signal components for which the ear is less sensitive to distortion. The second effect is the masking of a low amplitude tone (or noise) by another nearby higher amplitude tone. Both of these effects are explained in the following sections.

2.2.1 Masking Threshold in Quiet

A plot of the hearing threshold in quiet is shown in the figure below. The frequency range the ear may also be divided into critical bands. A critical band is the bandwidth within which signal intensities are added to decide if the combined signal exceeds a masking threshold[7]. The frequency scale which is derived by mapping frequencies to critical bands is known as the Bark scale with 20 kHz equal to 24 barks. The figure below is the masking threshold of the ear in quiet plotted on a linear frequency scale. As can be seen in the figure, the ear is most sensitive to signals or noise in the midrange and least sensitive to signals in the lowrange or highrange.

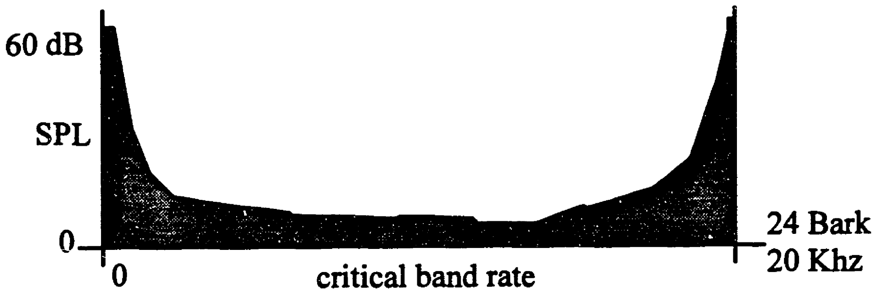

Figure 8 Masking Threshold in Quiet

The hearing threshold describes the minimum detectable sound pressure level of a single sinusoidal tine. For instance, at 2 kHz any sine wave above 5 dB is audible, while at 100 Hz and at 14 kHz the power of a sine wave must be greater than 25 dB to be audible. So the threshold of hearing obviously varies with frequency. Any input whose power is below the masking threshold is inaudible, irrelevant to the listener, and does not need to be coded.

2.2.2 Masking Effect of a Tone

While the previous section described how a tone may be masked by being below the hearing threshold of the ear in quiet, this section described how a tone may be masked by another louder tone at a nearby frequency. This effect is shown in the figure below.

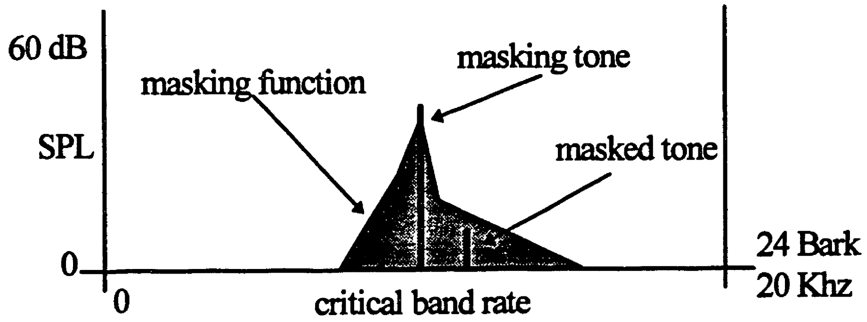

Figure 9 Masking Effect From A Tone

The masking effect is due to mechanical processes in the Cochlea[7]. If the tone is below the masking threshold from the masking tone, then the tone is inaudible and does not need to be coded. The specific shape of the masking function depends both on the frequency (in Hz) and the amplitude of the masking tone. In the linear frequency scale, the masking effect will be narrow at low frequencies and wide at high frequencies.

2.2.3 Total Masking Threshold

To determine the total masking threshold, the masking threshold in quiet is added to the masking threshold in the presence of tones. The result is the total threshold, which gives the minimum audible signal versus frequency. The total threshold can be used to determining if another signal (a tone or noise) is audible. The threshold is especially useful in determining if the noise introduced by quantization will be audible.

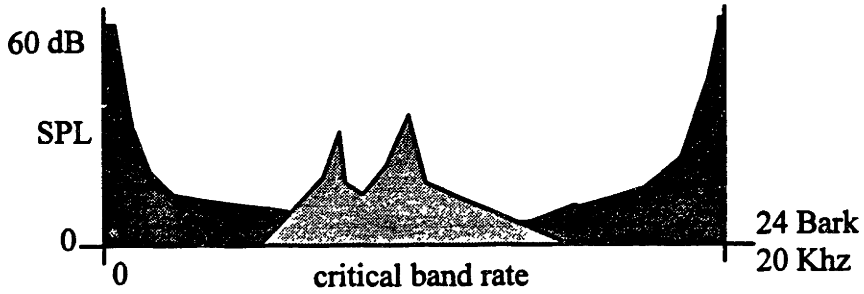

Figure 10 Total Masking Threshold

The purpose of the psychoacoustic model is to determine the optimum bit allocation to keep the noise introduced by quantization below the masking threshold. The output of the psychoacoustic model is the signal-to-mask ratio, the difference between the signal power in the band (in dB) and the power of the masking threshold (in dB). This ratio is necessary for determining if the quantization noise is below the masking threshold. After a discussion of the quantization of samples in the next section, the relationship between quantization noise and the masking threshold will be more clear.

2.3 Representation Of Samples

The input to the MPEG encoder is a stream of sixteen bit signed audio samples in the range [-32768, 32768]. To reduce the bitrate to transmit the signal, or to reduce the total number of bits required to store the signal, the samples are recoded with fewer bits per sample. This process is called quantization (even though the input samples are already quantized) since the samples are quantized to different levels. To assure that the samples occupy the full range of the quantization levels, the samples in each subband are also normalized and scaled. The scaling and quantizing processes are described in the following sections.

2.3.1 Scaling

The first step in scaling the integer input samples is to normalize them from the possible range of [-32768, 32768] to [-1,1]. It is possible that the samples only occupy a portion of this range. To maximize the range of the samples, the value of the largest sample in each subband is found; this value is known as the scalefactor. When all of the samples in the range are divided by the scalefactor, the samples will fill the entire range [-1,1].

In layer I and II a scalefactor table gives 63 scalefactors, each of which is identified by an index number. In layer I, once the largest of these 12 samples is found, the value is compared to entries in the scalefactor table. The scalefactor chosen for the subband is the entry in the table with the next smallest value to the maximum sample. The table of scalefactors for layers I and II can be found in the MPEG specification in 3-Annex B, Table 3-B.1.

In layer II there are 36 samples in each subband per channel. Like layer I, a scalefactor is found for each group of 12 samples. If the any of the three scalefactors are similar, only one or two scalefactors are used to scale two or three groups of samples. To determine how many scalefactors are needed, the difference in index numbers is found between the first and second, and the second and third scalefactor. Depending on the two differences between the indexes either one, two, or three scalefactors are used and transmitted. For example if the difference in the first two scalefactors is greater than three and the difference between the second two is zero, then two scalefactors are transmitted, one for the first set of 12 samples and one for the other 24 samples. By using scalefactors that are not exactly matched to the range of the samples, the signal-to-noise ratio of the quantization noise is reduced, but fewer bits are required to transmit the bit allocation. A table in the MPEG-1 specification (Table 3-C.4) gives all of the specific transmission patterns.

The scalefactors range from 0.00000120 to 2.00. The minimum scalefactor value, equivalent to -120 dB, determines the power of the quietest signal that can be represented using the full range of the quantizer. This value is chosen to fit the dynamic range of the ear[8]. The scalefactors increase in 2 dB increments up to 6 dB. The size of the increment determines the how much of the full range of the quantizer is used (the sample range can only be matched to [-1,1] in every case if the step size between the quantizers in infinitely small). So we can see that if less than three scalefactors are used in layer II, the samples may occupy up to 8 dB less than the full range of the quantizer (there is 2 dB slack with the original scalefactor with an addition 2 dB for each difference in index). Since we normalized the samples to the range [-1,1] it may seem wasteful to have several scalefactors greater than one. However, the samples are normalized before the analysis filter, and there are some cases where the output samples could have a magnitude greater than one (i.e. if the input samples have a magnitude of one everywhere and have the same sign as the analysis filter impulse response samples).

2.3.2 Quantizing

Each of the samples input to the encoder is coded with 16 bits. To reduce the amount of information that must be stored and transmitted to the decoder, the samples may be coded with a variable number of bits (between 0 and 16). The number of bits to encode the samples in any subband is determined by the psychoacoustic model as described in the previous section.

Although the samples input to the encoder have already been quantized (for instance by an analog to digital converter), the samples are quantized again with a lower number of bits. The number of levels, L, available for quantization is equal to 2^{nb}, where nb is the number of bits in the new quantization. In one method of performing the quantization, the scaled samples in the range [-1,1] are normalized to [0,L-1] by multiplying the samples by L/2 and translating the samples by L/2. The nb most significant bits are retained in the calculation. The process of retaining the nb bits is equivalent to truncating the samples to the nearest integer number.

At this point the samples are represented by three numbers: the scalefactor, the bit allocation, and the quantized sample value. The quantized samples may be described by the following formula

$$Quantized\ Sample = RND\left(2^{nb-1} \times (\frac{Sample}{ScaleFactor}) + 2^{nb-1}\right) \qquad (10)$$

The error induced by quantizing the samples, known as the quantization error, is

$$e[n] = x'[n] - x[n] \qquad (11)$$

where x'[n] is the quantized sample and x[n] is the original.

Since the samples are quantized by truncating the sample down to the nearest quantization level, the error for each sample is in the range

$$-\Delta < e[n] < 0 \tag{12}$$

Where Δ is the step size of the quantizer. Since we have normalized the input samples to the number of quantization levels, the step size Δ is equal to one. If the step size is small compared to the total range of the quantization levels, then it is reasonable to assume that the error is uniformly distributed between minus one and zero. The variance under these assumptions is

$$\sigma_e^2 = \frac{\Delta^2}{12} = \frac{1}{12} \tag{13}$$

A common measure of the degradation of a signal by additive noise is the signal-to-noise ratio, defined as the ratio of the variance of the signal (after the signal has been normalized to the number of quantization levels) to the variance of the noise. The signal-to-noise ratio for an (nb) bit quantizer is[9]

$$SNR = 10\log_{10}\left(\frac{2^{nb-1}\sigma_x^2}{\sigma_e^2}\right) \tag{14}$$

$$SNR = 6.02(nb-1) + 10.8 - 20\log_{10}\left(\frac{1}{\sigma_x}\right) \tag{15}$$

Thus the signal-to-noise ratio increases linearly with the number of bits used to quantize the samples. It is the goal of the encoder (and specifically the psychoacoustic model) to keep the quantization noise below the threshold of hearing of the ear. The signal-to-mask ratio and signal-to-noise ratio can be used to determine if the noise introduced by quantization in a subband is inaudible. The noise will be inaudible if the power in the quantization noise is less than the power of the masking threshold. Since the signal-to-noise ratio is the power in the signal minus the power in the noise and the signal-to-mask ratio is the power in the signal minus the power in the mask, the noise will be inaudible if the signal-to-noise ratio is greater than the signal-to-mask ratio. If it is determined that the noise in a subband is audible, then the number of bits used to quantize the signal in that subband should be increased.

2.3.3 Quantization Example

Let us calculate the number of bits saved by quantizing the samples in a mono 48 kHz, 128 bit/s layer II stream. As a simple example, assume that the first six subbands are coded by eight bits, the next six subbands by four bits, and the remaining subbands by zero bits. Each frame in layer II contains 1152 samples. At 16 bits/sample the original samples requires 18 Kbits/frame. Also since each frame is 24 ms long the samples require 768 Kbits/sec, well above the maximum bit rate per channel of 192 Kbits/sec.

The quantized samples require 6*36*8 + 6*36*4 = 2592 bits/frame or 108 Kbits/sec. Thus quantized signal is represented by 660 Kbits/sec fewer bits per second. However the initial calculation does not take into account the number of bits required to transmit the scalefactor or bit allocation information (necessary to decode the compressed signal). This overhead information will be described in more detail in a following section, but it will add at least 312 bits, increasing the total bit rate to 121 Kbits/sec. The total compression with the specifications given above works out to 6.3:1.

2.3.4 Decoder

In each frame, the decoder receives three sections of information: the scalefactors for each subband, the bit allocation for each subband, and all of the quantized samples in every subband. The decoder must reconstruct the original samples (plus some error value) from the above information.

To reconstruct the samples, the inverse of the encoder operations is performed. The quantized samples are shifted by L/2 ($L= 2^{nb}$) where nb is the number of bits in the bit allocation for the subband. This operation shifts the samples to the range [-L/2, L/2-1]. Then the samples are denormalized by the scale value, L/2 so the samples are mapped to the original range [-1,1]. While the range of the samples is now the same as the original samples (the samples in the encoder before quantization), the samples within that range still have a discrete set of values (in fact there are exactly 2^{nb} levels). The samples are then rescaled by multiplying by the transmitted scalefactors. Before the samples are renormalized to integer values for output, some additional processing (such as dematrixing) may take place.

2.4 CRC Error Checking

The cyclic redundancy check (CRC) is an error detection technique used in the MPEG codec. A protection word is generated from the audio data in the encoder and transmitted in the bit stream. In the decoder, the same bits are used to calculate another protection word which is compared with the transmitted CRC protection word. If the two words are different, then an error occurred in transmission.

2.5 Multichannel Configuration

The MPEG-2 Audio encoder may accept as many as six channels as inputs or as few as one channel. The number of different channels and their placement in a room is known as the configuration. When all six channels are used, there will be three front channels (Left, Right and Center), two surround channels (Left Surround and Right Surround), and a low frequency channel. Since the low frequency channel is not a full channel (it has a bandwidth of 125 Hz), the configuration with all six channels is also known as the 3/2 plus LFE format. The first number refers to the number of front channels and the second number refers to the number of surround channels.

With its multichannel capability, MPEG-2 allows for audio programs to be transmitted simultaneously in different languages. The 3/0 + 2/0 configuration has Left, Center, and Right channels of the first program and the Left and Right channels of the second program. A fully functional MPEG-2 decoder will decoder and output all five of these channels; the audio system should allow the user to select which audio program he/she would like to here. A table with a definition of all of the configurations is shown below.

Number Of Channels	*Configuration*	*Description*
5	3/2	Front channels L, C, R and surround channels LS and RS
5	3/0 + 2/0	L,C,R of first program plus L2,R2 of second program
4	3/1	L, C, R and mono surround sound channel S
4	2/2	L, R and LS,RS
4	2/0 + 2/0	L,R of first program plus L2, R2 of second program
3	3/0	L,C,R with no surround
3	2/1	L,R with mono surround S
2	2/0	L,R
1	1/0	one mono channel

Table 1 Multichannel Configuration

The definitions shown in Table 1 are for the encoder and the decoder. The encoder should be able to accept data in any of the above formats, correctly store the data in the number of necessary channels, and record the configuration information for use in the decoder. Exactly how the configuration information is stored is explained in the MPEG-2 Implementation section. In addition to the encoder being able to accept data from any of the formats, the decoder should also be able to convert data to the configuration of the user's stereo system. If the data was encoded in the 2/2 configuration (L,R plus LS,RS) and the user has a 3/1 system (L,C,R plus S) then the surround signals may be combined into one mono surround signal and the Left and Right signals may be combined into a Center signal.

2.6 Matrixing/Dematrixing

The matrixing of audio channels provides backwards compatibility of a stream encoded with an MPEG-2 encoder to an MPEG-1 decoder. That is, an MPEG-1 decoder is able to decode and display the "basic stereo information"[2]. The basic stereo information may consist of a downmixed version of all of the channels or of just the left and right channels.

There are three unique matrixing equations available for use in the encoder. The first combines all of the channels in the configuration into the backwards compatible channels, the second assigns the left and right audio to the backwards compatible channels, and the third implements a Dolby Prologic encoding system. The equations for matrixing are given in Table 2.

Matrix Procedure	*Matrix Equation*
0,1	L_0=L+x*C+y*LS R_0=R+x*C+z*RS
2	L_0=L+x*C-y*jS R_0=R+x*C+y*jS
3	L_0=L R_0=R

Table 2 Matrix Equations

The terms in matrix equations 0,1, and 3 are divided by the weighted factors to give different emphasis on the front, center and surround audio channels. For all of the weights chosen, if the channels have unit amplitude, then the combined channels will have larger than unit amplitude. Thus, the amplitude of the combined signal is normalized by dividing by a denormalizing factor.

2.7 Transmission Channel Allocation

An important concept in the MPEG-2 audio system is the transmission channel allocation. In section 2.7 it was described how the channels were assigned in the channel configuration. The transmission channels are the structures that hold the audio data for the different audio channels when the data is transmitted between the encoder and the decoder. There are five full transmission channels plus a partial channel for the low frequency channel.

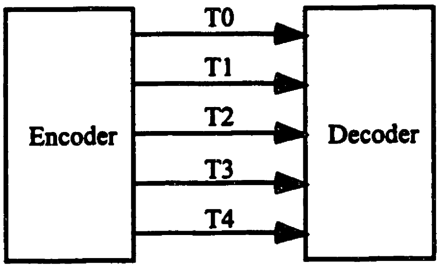

Figure 11 Transmission Channels

The first two channels, T0 and T1, always hold the MPEG-1 compatible channels L0 and R0. The three remaining transmission channels hold three of the five channels L, R, C, LS, and RS. The channels T0 and T1 are transmitted in the MPEG-1 compatible part of the frame and the channels T2, T3 and T4 are transmitted in the multichannel extension part of the frame. The two audio channels not transmitted explicitly are derived from L0 and R0 using the matrixing techniques described in the previous section.

If the encoder is using a 3/2 stereo format, the three audio channels may be assigned in several different ways to the three available transmission channels. The transmission channel T2 will always contain the center channel if it is explicitly transmitted otherwise it will contain the left or right channel. T3 will contain either the left or left surround channel. T4 will contain either the right or right surround channel. Table 3 shows the available transmission channel allocations for the 3/2 configuration.

Channel Allocation	*T2*	*T3*	*T4*
0	C	LS	RS
1	L	LS	RS
2	R	LS	RS
3	C	L	RS
4	C	LS	R
5	C	L	R
6	R	L	RS
7	L	LS	R

Table 3 Transmission Channel Allocation

The MPEG-1 compatible channels L0 and R0 may be used to predict the channels T2-T4 with the multichannel prediction function. Prediction is most effective if the predicted channels contain a minimum amount of information (in fact prediction would be perfect and trivial if the predicted channel contained no information). For this reason, the channels with the least information are assigned to channel T2-T4. Assigning the channels with the least information to T2-T4 will also make dynamic crosstalk more efficient for the same reasons.

2.8 Dynamic Crosstalk

Dynamic Crosstalk is the multichannel extension of intensity stereo coding, a method used in MPEG-1 to improve quality at low bitrates. In intensity stereo coding, instead of transmitting separate left and right subband samples only the sum of the signal is transmitted, but with scalefactors for both the left and right channels, thus preserving the stereophonic image.[1] This maintains the stereophonic image because at high frequencies, the listener determines the source of a sound by the temporal envelope and not by the temporal fine structure of the audio signal. The channels still use their original scalefactors, so the intensity information is retained. The fine structure of each channel has been changed by adding information from another channel, but does not affect the localization of the sound due to the psychoacoustic result described above.

The extension to multichannel encoding is that any of the multichannels (T2-T4) may be coded in dynamic crosstalk mode with any other transmission channel. As in intensity stereo coding, the dynamic crosstalk channel uses samples and bit allocation from another channel, but has its own scalefactors. The figure below shows how the use of a different set of scalefactors would change the signal in one subband of a channel.

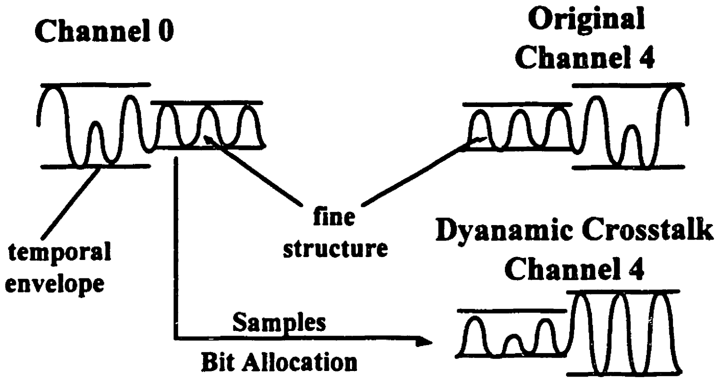

Figure 12 Dynamic Crosstalk Encoding

Most of the added complexity for dynamic crosstalk is in the encoder. An algorithm must be developed to determine when dynamic crosstalk will successful (and effective) and between which channels it should be used. This is an easier problem in MPEG-1, because intensity stereo is only possible between the left and right channels. In MPEG-2, there are five channels with many different possible configurations. The encoder must be able to decide when dynamic crosstalk is necessary and then evaluate between which channels it will be most effective.

In the decoder, the channels encoded with dynamic crosstalk must be reconstructed. If dynamic crosstalk is enabled in a channel and in a certain subband group, then the bit allocation and the coded subband samples are missing. They have to be copied from the transmitted subband samples of the corresponding channel. The scalefactor selects and scalefactors for both channels are transmitted in the bit stream. If the bit allocation and the samples are copied first, then the samples can be reconstructed just like any other channel. The dynamic crosstalk reconstruction operation is shown in figure below.

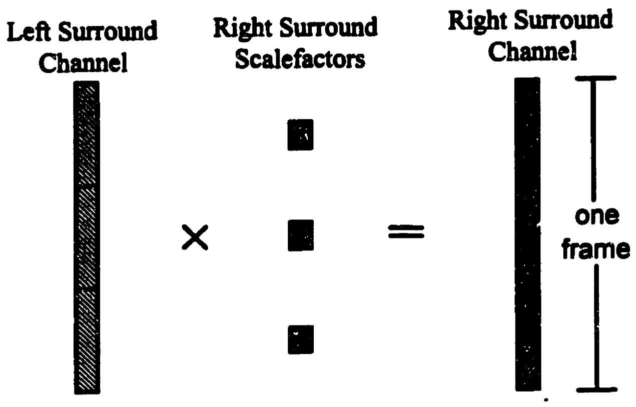

Figure 13 Dynamic Crosstalk Calculation

2.9 Adaptive Multichannel Prediction

The objective of multichannel prediction is to reduce statistical redundancies between channels. To eliminate the statistical dependencies, one channel is used to predict another. In prediction, the original channel is transmitted along with a prediction coefficient (to predict the second channel), and error signal. The error signal is the difference between the second channel and the estimate of that channel produced from the original channel. The prediction problem may be carried out in the encoder so that the error signal is orthogonal (although not statistically independent) to the original signal.

The encoder must determine between which channels multichannel prediction will be effective. One computationally intensive but effective way to do this would be to calculate the power in the error signal resulting from all combinations of prediction channels. The prediction can be carried out by some technique such as Deterministic Least Squares.[10] The channels for which prediction yields the lowest power error signals should be chosen.

In the decoder, channels encoded with prediction must be reconstructed. Four pieces of information are required to reconstruct a channel: the prediction coefficient (the same prediction coefficient is used for all the samples in each subband for a whole frame), the delay value, the subband sample in T0 or T1, and the error signal. It is important that the original channel and the error signals be dequantized before carrying out the prediction. For first order prediction with no delay, the samples in channel T0 are multiplied by the prediction coefficient and the error signal is added to the result. The operation is shown in Figure 14 below.

For higher-order prediction with delay, the operation is only slightly more complicated. Up to three samples in the original channel are multiplied by prediction coefficients. The three samples are always consecutive. The samples used in the computation may be shifted from the predicted channel by up to seven samples. For instance, in second order prediction with a delay of five, samples one, two and three of channel T0 are multiplied by prediction coefficients and summed to get the predicted sample number eight. The error signal is added after the summation.

$$LS[8]=pred\_coef[0]*T0[3]+ pred\_coef[1]*T0[2]+ pred\_coef[2]*T0[1]+\varepsilon[8] \qquad (16)$$

Notice that for higher order prediction, prediction will use samples from the previous frame.

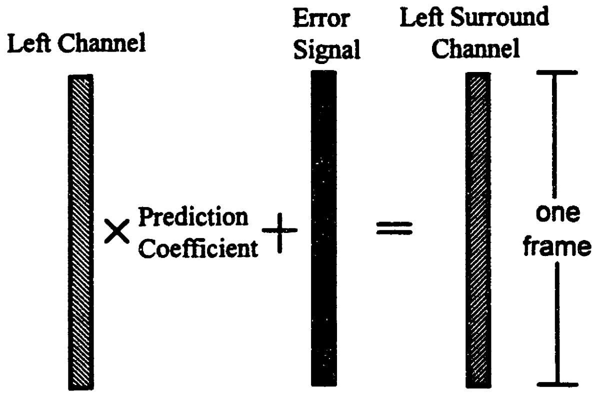

Figure 14 Multichannel Prediction

2.10 Phantom Center Coding

The purpose of the center channel in the MPEG-2 system is to provide a stable center sound image in "cinema-like" applications[7]. A technique called phantom center coding is used to reduce the bit rate of the center channel while preserving the stable center image. In phantom center coding, the high frequency parts are removed from the center channel and added to the left and right channels. The new left and right channels must create a phantom source of sound at the center channel. The bit allocation of subbands eleven and higher center channel is set to zero, no scalefactors or samples are transmitted for these subbands.

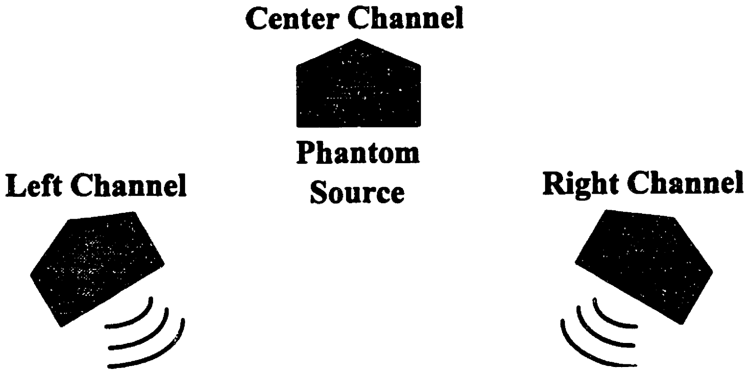

Figure 15 Phantom Center Coding

This function adds little additional complexity to the decoder; all that is required is a switch to set the bit allocations equal to zero. In fact it requires fewer cycles for decoding since part of the center channels does not have to be unpacked or dequantized. Additional complexity is added in the encoder since the encoder must implement an algorithm to decide when phantom center coding should be used and how the center channel should be distributed to the left and right channels.

Phantom center coding is more effective at high frequencies than at low frequencies since the phase at high frequencies is less important to the listener in determining the source of a sound. At lower frequencies, the wavelength of a sound wave is about the same as the dimension of a human head, so it is possible to determine the source of a sound by the phase difference of the signal at each ear. However at high frequencies the phase becomes less important with since the wavelength becomes smaller, eventually much less than the dimensions of the human head. Splitting the sound from the center channel into the two front channels will change the phase information received by the listener. Phantom center coding will only work if that phase information is irrelevant.

The intensity of sound is the other cue that is used by a listener to determine the source of a sound. A sound coming from the left of a person should be louder in that person's left ear than in their right ear. For phantom center coding to be effective, the intensity cues of the left and right channels must give listeners the impression that sound is still coming from the center. It is necessary to implement an algorithm in the encoder that effectively divides the center channel and adds the signal to the left and right channels.

2.11 Extension Bit Stream

2.11.1 Introduction

The MPEG-2 audio standard contains a provision which allows the use of an extension bit stream. The extension bit stream provides a way to increase the bit rate above the maximum rate defined in MPEG-1 (384 Kbits/s). If the extension bit stream is used, two audio bit streams are parsed out by the system decoder: the MPEG-1 compatible stream and the extension bit stream. Frames from the two streams are paired together increasing the number of bits to code the sample in each combined frame. Figure 16 below shows an MPEG frame with an extension bit stream.

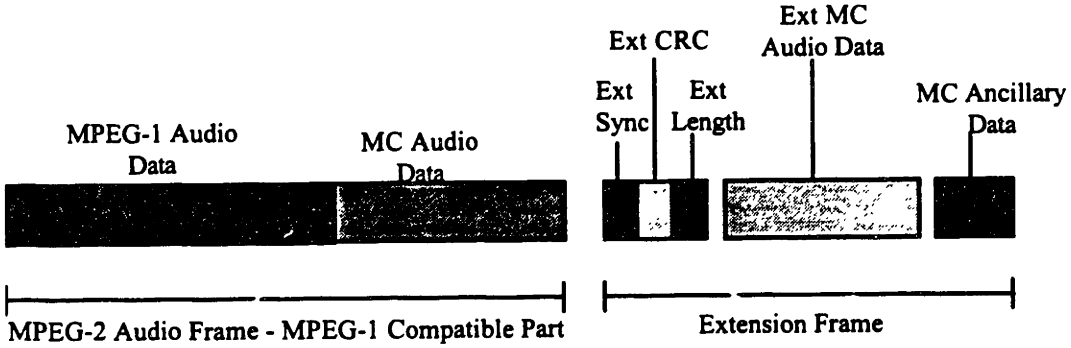

Figure 16 MPEG Frame with Extension Bit Stream

2.11.2 Extension Bit Stream Description

The extension bit stream consists of three parts: the extension header, the extension data, and the extension ancillary data. The extension header contains bits which describe the extension bit stream. The following values make up the extension header: the extension synchword, the extension CRC check, the extension length, and a reserved bit. The extension synchword is the twelve bit value, '0xfff'. The synchword is used to align the beginning of each frame so that the data fits correctly with the data in the MPEG-1 compatible frame. After the synchword is the extension CRC check, a 16 bit error checking word. Unlike the MPEG-1 CRC check, it must be utilized in every frame. The CRC check begins with the extension length bits and included a total of 128 bits. Following the CRC word is the 11 bit extension length value which defines the total length of the extension frame in bytes. The maximum value is, however, limited to 1536 bytes or 384 words. Since the duration of an MPEG-2 frame is 24 ms, the maximum additional bit rate is 512 Kbits/sec and the maximum total bit rate is 896 Kbits/sec. The use of the final element of the extension header, the reserved bit, has not been defined.

The extension length value defines the length of the entire extension frame, including the extension header and the ancillary data. To find the number of data bits in the extension frame, the number of bits in the header and in the ancillary section are subtracted the extension length. It is the extension data bits that must be synchronized with the data bits in the main stream.

The extension ancillary data field is the part of the extension stream that may contain ancillary data or hold padding bits. The padding bits and the ancillary data bits must be unpacked to bit accurately synchronize the next extension frame.

Several other values containing information useful to the decoding of the extension bit stream must be unpacked from the main data stream. A single bit value unpacked from the multichannel header indicates if an extension bit stream is used. If the bit is true, then another 8 bit value is unpacked, indicating the number of bytes that are used for the MPEG-1 compatible ancillary data field. The number is important when synchronizing the main data stream with the extension stream. The number of ancillary data bits is used to calculate with bit accuracy the end of the main data stream; the point where the unpacking should switch to the extension bit stream.

The extension bit stream contains the bits that would not fit into the MPEG-1 compatible bit stream. There are several important subtleties. First, the MPEG-1 compatible channels must be contained within the main bit stream because they must be decodable by an MPEG-1 decoder (the MPEG-1 decoder will not have extension bit stream capabilities). Second, the multichannel header must be contained in the MPEG-1 compatible frame. Since the header will signal the decoder that an extension bit stream exists, it is important the signal bit be contained in the main data stream. With these two restrictions, the two streams are filled sequentially with data bits. When the last bit of the main data stream is full, the next bits are put into the extension stream.

3. UNIX IMPLEMENTATION

3.1 Overview

The MPEG-2 audio decoder was implemented in the C programming language within a UNIX operating system. The functions implemented in the decoder include the functions to unpack the bit stream, to reconstruct the subband samples, to ensure MPEG-1 compatibility, and to decode dynamic crosstalk and multichannel prediction.

Many of the functions in the MPEG-2 decoder are the same as an MPEG-1 decoder. Namely, the unpacking functions for the header, scalefactor selects, scalefactors, and subband samples for channels T0 and T1 are identical to the functions in an MPEG-1 decoder. The functions for requantizing, rescaling, and renormalizing the samples are also similar except they operate on up to five channels and must deal with samples encoded with dynamic crosstalk and prediction. A discussion of an MPEG-1 audio decoder is included to give the reader background information on how the MPEG audio system operates. The MPEG-1 audio decoder described was implemented by Jon Rowlands of Texas Instruments, Inc. The functions specific to an MPEG-2 decoder are covered in more detail beginning in section 3.3. In addition, the performance of the MPEG-2 decoder is examined in results section.

3.2 MPEG-1 Implementation

The UNIX implementation of the MPEG-1 audio decoder algorithm has the following features:

- 16 bit output sample resolution
- 32, 44.1, 48 kHz output sample rate
- 32 to 384 Kbits/s input bit rate

- 3:1 to 24:1 compression
- mono, stereo, joint stereo or dual channel
- MPEG layers 1 and 2
- robust against digital channel errors
- self-configuring based on input data

Figure 17 below shows a block diagram of the functions required in an MPEG-1 decoder. The input to the decoder is the encoded bit stream. The bit stream is broken into 8 ms or 24 ms frames. From each frame, the decoder unpacks the frame header, bit allocation, scalefactors, and subband samples. The header information is used to determine the properties of the frame (bit rate, sample rate, etc.) and how to unpack and decode the other sections. When all the sample information in the frame has been unpacked, the samples are requantized and rescaled. Finally, the subband synthesis filter transforms the subband samples back into a full bandwidth signal. The processes for unpacking the samples and reconstructing the audio signal are explained in the following sections.

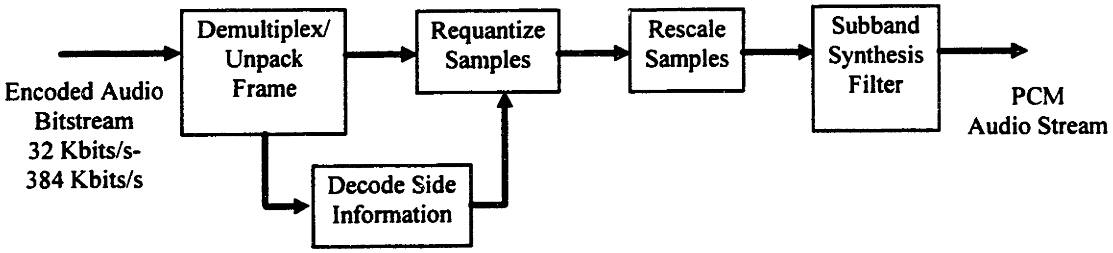

Figure 17 MPEG-1 Decoder

3.2.1.1 MPEG-1 Header

The MPEG header contains both a synchronization key and information about the structure of the rest of the MPEG frame. Among other things, the header indicates which layer of encoding is used, the bit rate, the sampling rate and the sampling frequency. The interpretation of these important pieces of data is explained below.

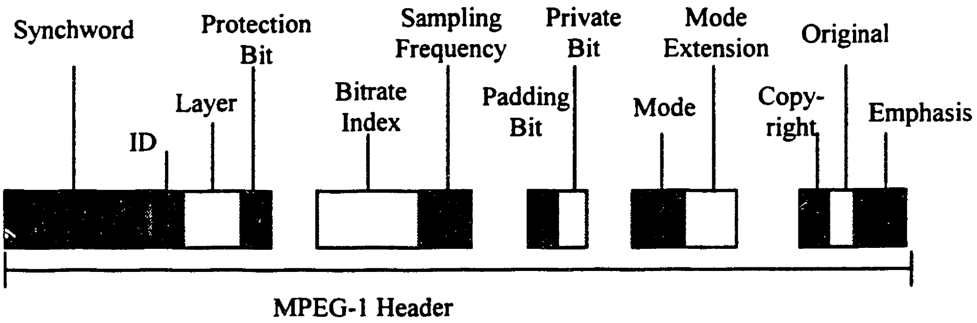

Figure 18 MPEG-1 Header Bit Stream

The header contains a synchronization word, the twelve bit string '1111 1111 1111' which is used to synchronize the decoder on the start of a frame. The decoder must search through the audio data to find the synchronization word at the beginning of the stream and also if it must re-synchronize because of an error.

As the ID bit was defined in the original MPEG-1 specification, '1' meant that the frame was MPEG audio and '0' was a reserved value. The reserved value is defined in the MPEG-2 specification to indicate that a different set of sampling frequencies is used. These lower sampling frequencies are used in low bitrate coding. The layer bit indicates the layer of encoding (layer I, II, or III) used for the frame.

The protection bit indicates whether a CRC check word is used to detect transmission error. If the bit is '0' then a sixteen bit CRC check word is included directly after the header. It may be desirable to turn the CRC check off if the quality gained from using the extra bits for coding is more important than knowing if an error has occurred.

The bit rate index is used to identify the bit rate of the stream. The bit rate may vary between 32 Kbits/s and 448 Kbits/s, but not all bit rates are valid for all modes. For example a layer I stereo stream may be transmitted at 256 Kbits/sec but not a mono stream (since the maximum bit rate per channel is 224 bit/s in layer I, 192 Kbits/sec in layer II, and 160 Kbits/s in layer III).

The three sampling frequencies allowed in the MPEG-1 specification, 44.1 kHz, 48 kHz are 32 kHz, are indicated by the values '00', '01', and '10'. If the ID bit was set to '0' then the corresponding sampling frequencies would be 22.5 kHz, 24 kHz, and 16 kHz.

The duration of a frame is determined by the sampling frequency and the number of samples in the frame. The number of samples in a frame is 384 in layer I and 1152 in layer II. The duration of the frame is the number of samples in the frame divided by the sample rate. Thus the length of a layer II frame with a sampling rate of 48 kHz is 24 ms. If the sampling rate is 44.1 kHz or 22.5 kHz then the number of bits in each frame is not a whole number. To adjust the average length of an audio frame in time to the mean bitrate, an extra slot (eight or thirty-two bits) may be added to the frame. An extra bit is added if the padding bit is a one.

The mode bits indicate the whether the encoded signal was mono, stereo, or a multilingual program. In a multilingual program, the two channels could be recorded in a different language. The mode also indicates if a stereo program has been encoded with joint stereo. The subbands where joint stereo is used are indicated by the mode extension variable.

The remaining bits indicate whether the stream has been copyrighted, if the stream is an original, and if any kind of emphasis was used in encoding the data. More information about these sections is included in the MPEG-1 specification.

3.2.1.2 Unpacking Audio Data

With the header unpacked and the structure of the frame determined, the decoder now has enough information to begin unpacking the audio data. The audio data sections that must be unpacked in MPEG-2 are the bit allocation, the scalefactors, and the quantized samples. The next sections describe the process for unpacking the audio data.

3.2.1.2.1 Unpacking Bit Allocation

The next section of data to be unpacked from the bit stream contains the bit allocation for the first two channels of data. The bit allocation indicates the number of bits that are used in encoding the samples in each subband. It is important to unpack and store the bit allocation even when the bit allocation is zero since it is important in determining what other information will be transmitted for that subband. For instance, the function for unpacking the scalefactors and subband samples will only unpack data in a subband if the bit allocation is non-zero. If the audio signal is mono, then only one bit allocation is unpacked for each subband. If the signal is stereo, then the bit allocation for each subband is interleaved by channel. The flowchart below in Figure 19 shows the algorithm for unpacking the bit allocation data for the first two channels of audio data.

The value unpacked for each subband is an index into a bit allocation table in the MPEG specification[11]. The table indicates how many bits are actually used to encode the sample. In layer I, the bit allocation itself is always four bits. However, in layer II the number of bits in the bit allocation varies by subband. More bits are dedicated to coding the bit allocation for low and middle frequency subbands since the ear in more sensitive in these ranges. Increasing the possible number of bit allocations allows the encoder to keep the quantization noise below the threshold without wasting bits.

There are several different bit allocation tables used in layer II; exactly which bit allocation table to use is determined by the sampling rate and the bit rate. Some subbands, especially higher frequency subband transmitted at low bit rates, may never have any bits allocated. For instance, at a sampling rate of 48 kHz, and a bit rate of 32 Kbits/s, only the first eight subbands may have bits allocated to them.

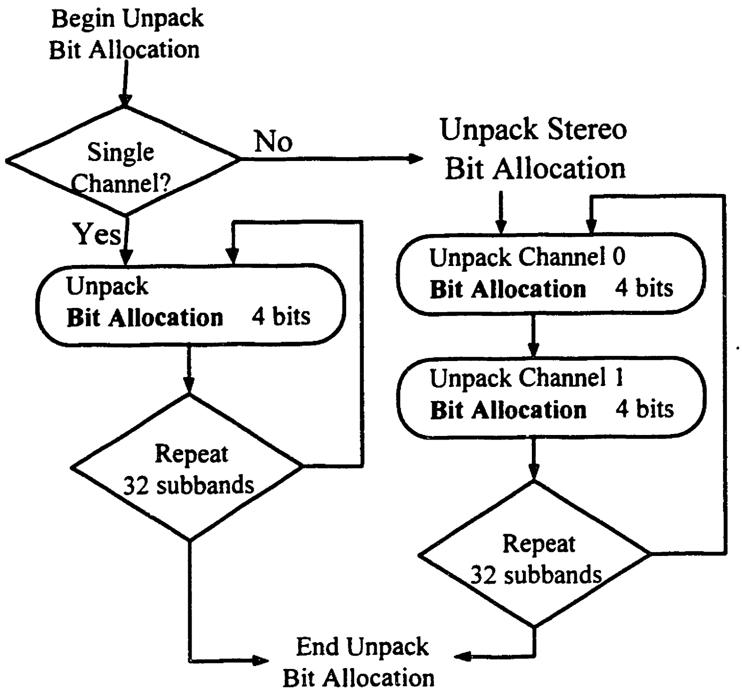

Figure 19 Bit Allocation Unpacking Flowchart

3.2.1.2.2 Unpacker Scalefactor Selects and Scalefactors

The next section in the frame contains the scalefactors and some information necessary to unpack them. In layer I there is one scalefactor transmitted for each allocated subband of each channel. Since layer II frames are three times as long (36 samples), three scalefactors are used for each subband. As explained in section 2.3.1, a scalefactor select for each subband indicates how many scalefactors are transmitted. The table below describes how the scalefactor selects are interpreted.

Scalefactor Select	Interpretation
'00'	3 scalefactors transmitted for parts 0,1 and 2 respectively
'01'	2 scalefactors transmitted. The first for parts 0 and 1 and the second for part 3
'10'	1 scalefactor transmitted valid for al three parts.
'11'	2 scalefactors transmitted. The first for part 0 and the send for parts 1 and 2.

Table 4 Scalefactor Select Interpretation

The scalefactor selects do not exist in a layer I frame. In a layer II, the scalefactor selects are unpacked first to determine the number of scalefactors transmitted in each subband. The unpacking of the scalefactors differs significantly between layer I and layer II. In layer I the scalefactors are interleaved by channel in each subband. In layer II the scalefactors are also transmitted by subband, but in each subband, a group of one, two or three scalefactors is transmitted for each channel (the number of scalefactors depends on the scalefactor select).

3.2.1.2.3 Unpacking Subband Samples

The structure and the unpacking of the subband sample section also differs significantly between layer I and layer II. In layer I, twelve samples are transmitted in each subband of each channel. Sample zero in each subband and channel is transmitted first, then sample one, and so on. The samples are interleaved by channel in each subband and are only transmitted if the bit allocation for that subband is non-zero.

In layer II, the frame is three times as long so there are three times as many samples per subband. The samples are transmitted the same way as layer I except three samples persubband per chann

II mono frame would be samples zero, one and two of subband zero. Next would be sample zero, one and two of subband one. The three samples may be grouped together into one value for transmission (the samples will be grouped if the number of quantization levels is three, five, or nine). The group must be split into the original sample values by a sequence of division and remainder operations. The specific algorithm for decoding the grouped samples is given in the MPEG specification.[12]

3.2.2 Reconstructing Audio Samples

Once the audio data has been unpacked, the samples must be requantized and rescaled. At this point the samples will be ready to be filtered back to the original full-spectrum time domain signal. If the decoder is decoding an MPEG-2 multichannel stream, then the dequantizing and re-scaling operations are carried out after all of the multichannel information is unpacked.

3.2.2.1 Dequantizing Subband Samples

The subband samples unpacked in the decoder are coded with between two and sixteen bits. The decoder must dequantize these samples by restoring a sixteen bit representation so that the full bandwidth signal may be reconstructed in the synthesis filter. The samples after dequantization still only have a discrete number of levels defined by number of coding bits during transmission.

The unpacked subband samples can be interpreted as integers each with $L=2^{nb}$ possible levels. To perform the dequantization, the quantized samples in the range [0,L-1] are normalized to the range [-1,1] by shifting the samples by L/2 (the shift value) and dividing the samples by L/2 (the scale value). The result, a floating point number in the range [-1,1], is the dequantized sample.

3.2.2.2 Rescaling Subband Samples

The samples are now floating point numbers, where all subbands have been scaled to fill the range [-1, 1]. The samples must be restored to their original amplitudes by multiplying by a scalefactor. The maximum amplitude of the samples after rescaling is still one, with many of the subband reduced to a lower amplitude.

The dequantizing and rescaling operations may be made more efficient by combining the operations. Since the same quantizer and scalefactor are used for groups of twelve samples, the quantizer itself may be scaled instead of the individual samples. This operation reduces the number of multiplications by nearly a factor of two. This operation for reconstructing the samples is shown in the equation below.

$$Sample = \frac{Scalefactor}{2^{nb-1}}(Quantized\_Sample - 2^{nb-1}) \qquad (17)$$

3.2.3 ISO Implementation of the Subband Filter

In the subband synthesis filter, the subband samples are interpolated by increasing the sample rate by a factor of 32 and filtering by the appropriate subband filter. The 36 samples in each subband each produce 1152 output samples. The output samples from each subband are summed together to produce the full bandwidth time domain signal. The figure below shows the synthesis operation.

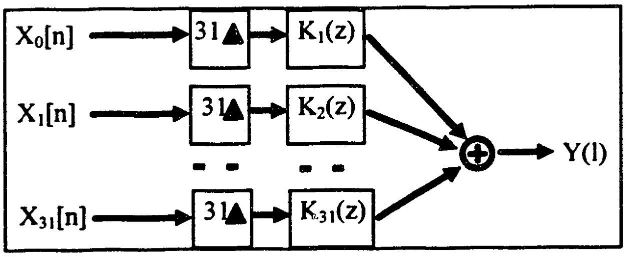

Figure 20 Subband Synthesis Filter

3.2.3.1 Implementation

The output of one of the filters is the convolution of the time domain representation of the filter with the subband signal.

$$Y_i[l] = \sum_{j=0}^{511} k_i[j] \cdot \hat{X}_i[l-j] \qquad \text{l=0...1152} \tag{18}$$

Where the sampling rate of the subband signals is first increased by 32 by inserting 31 zeros in-between the samples.

$$\hat{X}_i[l-j] = \begin{cases} X_i[m] & l-j = 32m \\ 0 & Otherwise \end{cases} \qquad \text{i=0...31} \quad \text{j=0...511} \quad \text{l=0...1152} \tag{19}$$

The filter k_i for each subband is the prototype filter multiplied in the time domain by a sinusoid to modulate the filter to the center frequency of the subband.

$$k_i[j] = m_i[j]p[j] \qquad \text{i=0...31} \quad \text{j=0...511} \tag{20}$$

$$m_i[j] = \cos\left((2i+1)(16+j)\frac{\pi}{64}\right) \qquad \text{i=0...31} \quad \text{j=0...511} \tag{21}$$

The complete signal is the sum of the output from each subband filter.

$$Y[l] = \sum_{i=0}^{31} \sum_{j=0}^{511} k_i[j] \cdot \hat{X}_i[l-j] \tag{22}$$

The filtering operation can be simplified by eliminating operations on the zero samples and by using the symmetry properties of the cosine function[13] [14].

3.2.3.2 Alias Cancellation

The filters were designed so that the aliasing components introduced into the subbands by decimation would cancel in the synthesis filter bank. The filter banks only have this ideal reconstruction property when there is no processing done on the signal between the filter banks. Processing subband signals by quantizing the samples adds quantization noise and degrades the cancellation of aliasing at the output. In an extreme example, suppose a subband is quantized with six bits, but the adjacent subband is turned off (zero bit allocation). There will be no signal in the unallocated subband so the aliasing components will not cancel. This situation would most likely only occur in two cases: first where there was no signal in the subband, and second where the bit rate is too low to adequately code all the subbands. The aliasing may be audible in either of these cases, depending on the filter banks and on the masking threshold.

3.2.4 MPEG-1 CRC Implementation

3.2.4.1 Selecting the bits to send to CRC

The bits used in computing the CRC word for the MPEG-1 compatible frame depend on whether the stream uses Layer I or Layer II coding. The header and bit allocation are protected in both layers as well as the scalefactor selects in Layer II. These are the only sections included in the CRC computation since they determine the structure of the frame. An error in one of these sections is catastrophic because it not only results in an error in the value unpacked, but can also cause further errors in the scalefactors and samples. For instance, if there is an error in the bit allocation, then samples of the wrong size will be unpacked. The value of these samples will be incorrect, as will following samples since the data in the stream has been shifted.

In layer I, both the number of bits used to code the bit allocation and the number of subbands is constant. The number of bits used to compute the CRC check is:

$$\text{Protected Bits} = \text{Header Bits} + \text{Bit Allocation Bits} \qquad (23)$$

$$\text{Protected Bits} = 16 + (\text{number Of Channels}) * (32 \text{ subbands}) * 4 \text{ bits} \qquad (24)$$

In Layer II, the number of transmitted subbands and the number of protected bits is variable. The number of protected bits in the header remains the same. However, the number of protected bits in the bit allocation changes depending on the bitrate and the sample rate. The number of bits in the bit allocation can be derived from the bit allocation tables in the MPEG specification and is shown below in Table 5. The number of bits in the scalefactor selects depends on the bit allocation since no scalefactor selects are transmitted for a null bit allocation. The total number of protected bits is the sum of the bits in the header, the bit allocation, and the scalefactor selects.

Bit Allocation Table Number	# of Protected Bits Bit Allocation Single Channel	# of Protected Bits Bit Allocation Stereo Modes
3-B.2a	142	284
3-B.2b	154	308
3-B.2c	42	84
3-B.2d	62	124

Table 5 Number of Protected Bits in Bit Allocation

3.2.4.2 Error Handling

If the transmitted CRC and the calculated CRC do not match than an error has occurred in a critical portion of the MPEG frame. Decoding of the current frame should stop, and the decoder should attempt to find the synchword of the next frame. However, some audio signal for the frame containing the error must be displayed. Two options are easily implemented: the output samples for the previous frame may be repeated, or the output may be muted by setting the samples to zero.

3.2.5 MPEG-1 Unpacker/Basic Unpacker Function

An important part of a decoder that is not described in the MPEG specification is the bit unpacker. The unpacker parses bits out of the bit stream and assigns the value of the bits to a variable. In the MPEG decoder, these variables include: the bit allocations, the scalefactors, and the quantized samples. It is important that the unpacker be efficient since it must be called every time a section of data is unpacked.

In preparation for a call to unpack bits, a structure must be defined for each variable that will be unpacked. Each structure contains an entry for the number of bits to unpack, and a pointer to the variable where the unpacked value will be assigned. In a call to unpackBits, an array of these structures is passed along with a structure which describes the data stream from which the bits will be unpacked. The data stream structure indicates the current word being unpacked in the stream, the next bit to be unpacked from the current word, and the location of the final word in the stream.

Since there is some overhead in calling unpack bits, the bit unpacker has been designed to unpack as many variables in one call as possible. For instance, the unpacker is able to unpack the header and all the subband samples in a single call. Since there are only twelve variables to be unpacked from the header, the efficiency in unpacking the header is limited by the overhead in the unpacker. Since there may be several hundred samples to unpack in a single call, the efficiency of the unpacker is limited not by the overhead, but by the efficiency of the section code which actually unpacks the bits.

3.3 MPEG-2 Implementation Overview

The MPEG-2 decoder has been implemented in the C programming language in the UNIX environment. An MPEG-2 frame may consist of one through five channels, the first two channels of which must be coded to be compatible with MPEG-1. Thus an MPEG-2 frame has two different parts, an MPEG-1 compatible part containing the first two channels, and another part containing channels 3-5, which we refer to as the multichannels. Like the MPEG-1 compatible part of the frame, the multichannel part also has associated information in a header and status section. This frame structure leads to a decoder with the following form: the first functions called in the decoder are responsible for unpacking the MPEG-1 compatible channels while the next group of functions are responsible for unpacking and decoding the MPEG-2 multichannel information. The flowchart for decoding the multichannel information is shown in Figure 21.

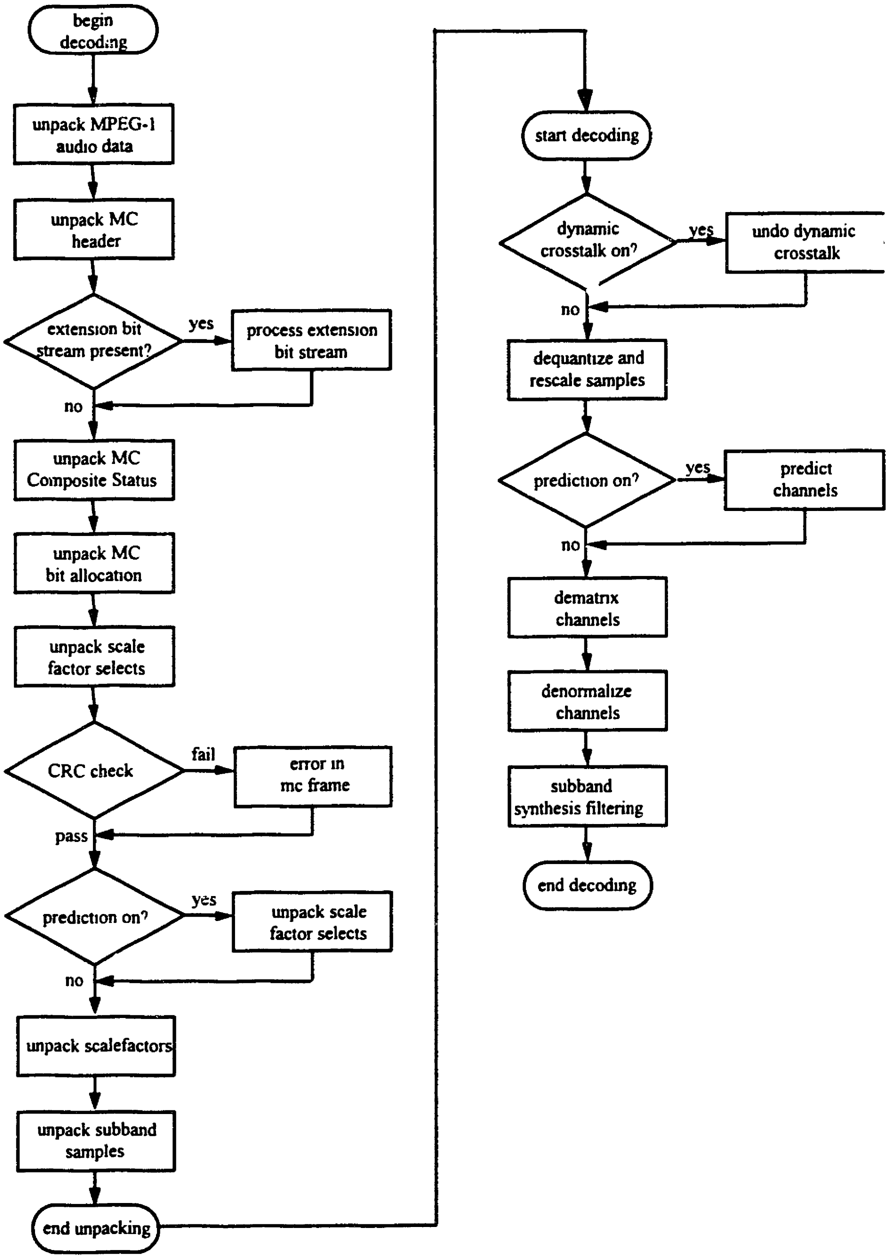

Figure 21 MPEG-2 Flowchart

As shown in the figure, the first step in the decoder is to unpack the MPEG-1 compatible parts of the stream: the header, bit allocation, scalefactors, and subband samples. Unlike MPEG-1, no further processing takes place on the stream until the multichannel data has also been unpacked. Next, the decoder unpacks the multichannel header and composite status sections. The fields in these sections provide information about the audio channel configuration, the transmission channel allocation, dynamic crosstalk, and multichannel prediction. The subsequent parts of the decoder unpack the bit allocation, scalefactors and subband samples as in MPEG-1, but also unpack an extra section containing more information about multichannel prediction. At this point in decoding, the data in the frame has been unpacked; the next parts of the decoder reconstruct the time domain samples all of the channels. The first step in restoring the samples is to processes the channels that have been encoded with dynamic crosstalk. Next, as in MPEG-1, the subband samples are requantized and rescaled. After rescaling, the information transmitted about prediction is used to generate the samples that were not transmitted. The dematrixed channels are also scaled to their original amplitudes by a denormalizing function. At this point, the missing samples have been reconstructed by dynamic crosstalk and multichannel prediction, and all of the samples are requantized and rescaled. Since the MPEG-1 compatible channels are a combination of several channels they must be dematrixed back into the original channels. Finally, just as in the MPEG-1 decoder, the channels are synthesis filtered to reconstruct the entire frequency spectrum and written to an output file.

3.4 MPEG-2 Implementation

The UNIX implementation of the MPEG-2 audio decoder algorithm has the following features:

- 16 bit output sample resolution
- 32, 44.1, 48 kHz output sample rate
- Supports MPEG-1 modes:
  - mono, stereo, joint stereo or dual channel
  - layers I and II
- Supports MPEG-2:
  - 3/2, 3/1,3/0,2/2,2/1,2/2,2/0,1/0 configurations

  - layer II
- MPEG-2 Functions:
  - transmission channel allocation
  - dematrix procedures 0, 1, and 2
  - dynamic crosstalk
  - multichannel prediction
  - phantom center coding
  - extension bit stream decoding
- MPEG-1 and multichannel CRC checking

3.4.1 Unpacking Multichannel Header

The first multichannel section to be unpacked is the multichannel header. The header contains the configuration of the encoded data, the matrixing procedure used in the encoder, the status of any multilingual programs, and the presence of an extension bit stream. The structure of the header is shown below in Figure 22. The next three sub-sections describe how the information in the header is interpreted to determine the configuration, the dematrixing procedure, and the multilingual status.

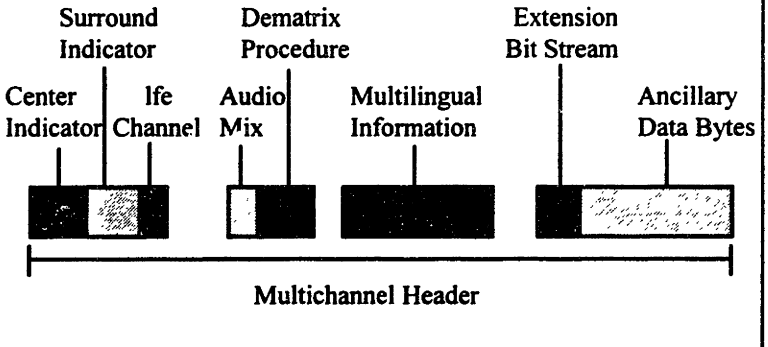

Figure 22 Multichannel Header Structure

3.4.1.1 Calculating the Configuration

The three values, Center Indicator, Surround Indicator, and LFE Present, unpacked from the bit stream determine the configuration and the number of multichannels in the system. The configuration was described in section 2.5 of the algorithm section. The number of multichannels are the number of channels transmitted in the frame in addition to the MPEG-1 compatible channels.

The center indicator is a two bit value which indicates whether a center channel is present and if the center channel is bandwidth limited. The bandwidth limited option is also known as 'phantom center' coding and was explained in section 2.10. The 3/2, 3/1, and 2/1 configurations all include a center channel. For the purpose of determining the configuration, the center indicator value which indicates phantom center coding is equivalent to the value which indicates a full center channel. If the center indicator is equal to '11', a variable called phantom, is set to true and the center indicator is reset to '01'. This center indicator value is used to determine the configuration.

The surround indicator value contains two pieces of information, the status of the surround sound channels and the presence of a second stereo program. The surround indicator can indicate the presence of zero, one, two surround channels. It may also indicate that there are now surround channels, but there is a multilingual program present. Like the center indicator, if the surround indicator is equal to '11', it is reset to '00' and another variable, mlStereo (for multilingual stereo) is set to true. This surround indicator value is used to determine the configuration.

The value LFE Present, simply indicates the presence or absence of a low frequency effects channel. The presence or absence of a low frequency channel does not change the definition of the configuration. A configuration with the low frequency effect channel present would be indicated by 3/2 + 1, instead of 3/2. The table below summarizes the interpretation of the center indicator, surround indicator, and LFE variables.

Variable	Value	Interpretation
Center Indicator	**'00'**	No Center Channel Present
	'01'	Center Channel Present
	'10'	Not Defined
	'11'	Center Bandwidth Limited
Surround Indicator	**'00'**	No Surround
	'01'	Mono Surround
	'10'	Stereo Surround
	'11'	No Surround, Second Stereo Program Present
LFE Present	**'0'**	No Low Frequency Effect Channel Present
	'1'	Low Frequency Effect Channel Present

Table 6 Configuration Information

The modified center indicator and surround indicator, along with the variable for the number of MPEG-1 audio channels are used as indexes into Table 7 of the system configurations. The number of audio channels variable is useful only when there are no multichannels present, then the variable determines if the MPEG-1 compatible data is mono or stereo. The table below does not include the configurations with multilingual programs since the multilingual information is now contained in the variable mlStereo.

centerIndicator	***'00'***				***'01'***		
surroundIndicator	***'00'***	***'00'***	***'01'***	***'10'***	***'00'***	***'01'***	***'10'***
# of AudioChannels	***1***	***2***					
configuration	1/0	2/0	2/1	2/2	3/0	3/1	3/2

Table 7 Channel Configurations

Once the center indicator and the surround indicator have been modified as described above, they may be used to directly calculate an important parameter for unpacking data and reconstructing samples, the number of multichannels.

numberOfMultiChannels = centerIndicator + surroundIndicator; (25)

The value center indicator is either 1 or 0 depending on the presence or absence of a center channel. The value surround indicator may be 0, 1, or 2 for no surround channels, a mono surround channel, or stereo surround channels. With the above possible values for center indicator and surround indicator the number of multichannels can range from 0 to 3.

3.4.1.2 Dematrixing Procedure

The next transmitted value in the multichannel header, the dematrixing procedure, indicates which matrixing procedure was used in the encoder and the corresponding dematrixing procedure to be used in the decoder. The dematrixing procedures that have been implemented are described in the algorithm section. Dematrixing procedure 2 is only valid with the 3/1 or 3/2 configurations; it requires a mono surround signal present in those configurations. The table below shows the interpretation of the dematrixing procedure value.

Dematrix Procedure Value	Interpretation
'00'	procedure 0
'01'	procedure 1
'10'	procedure 2
'11'	procedure 3

Table 8 Dematrix Procedure Values

3.4.1.3 Multilingual Information

Three variables in the header describe the multilingual portion of the data. Although these variables are unpacked from the header, no multilingual processing has been implemented in this version of the encoder.

The first multilingual variable, the number of multilingual channels is interpreted as an unsigned integer of three bits indicating the number of multilingual or commentary channels in the multichannel extension bit stream. The second variable, the multilingual sampling frequency indicates if the sampling frequency of the multilingual channels is the same as the main audio channels. If the value of the variable is zero, the sampling frequencies are the same. If the value of the variable is one, the sampling frequency of the multilingual channel is 1/2 that of the main audio channels.

3.4.2 Unpacking Composite Status

The composite status section of the MPEG-2 frame contains information about the transmission channel allocation, dynamic crosstalk, and prediction. The structure of the composite status information is shown below. The following three sections describe how to unpack and interpret the composite status information.

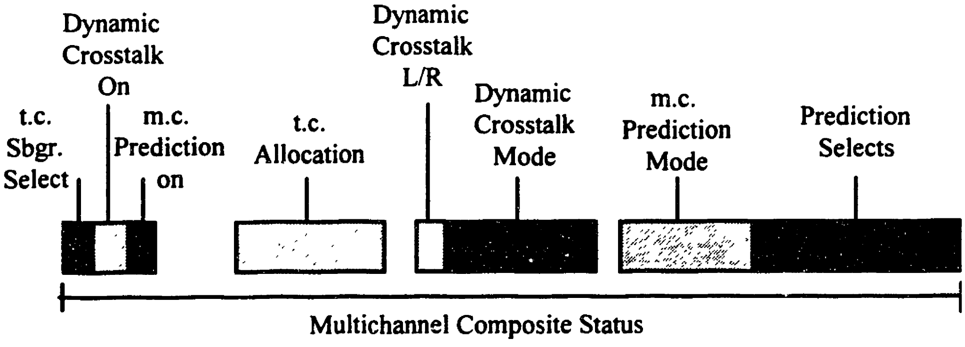

Table 9 Multichannel Composite Status Structure

The first three single bit values to be unpacked from the composite status section, indicate the mode of the transmission channel allocation, dynamic crosstalk, and prediction. The following sections describe how to interpret the first three bits and the further information that must be unpacked from the bit stream.

3.4.2.1 Transmission Channel Allocation

The five audio channels in the frame, L, R, C, LS and RS, can be assigned to any of the three transmission channels T2- T4. A table of the different transmission channel allocations for the 3/2 configuration is shown Table 12. Either a single transmission channel allocation may be transmitted for all of the subband groups, or a different transmission channel allocation for each group is determined by the one bit value transmission channel subband group select.

TC Sbgr Select Value	Interpretation
1	tc Allocation valid for all subband groups
0	tc Allocation valid for individual subband groups

Table 10 Transmission Channel Select Values

For some MPEG-2 information, the subbands are grouped together in to subband groups. For example, a separate transmission channel allocation may be transmitted for each subbands group. This reduces the amount of overhead information that must be transmitted, but also the flexibility of assignment to the different subbands. The following table shows the assignment of subbands to subband groups.

Subband Group	Subbands Included in Group
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8..9
9	10..11
10	12..15
11	16..31

Table 11 Subband Group Assigment

The flowchart for unpacking the transmission channel allocation is show in Figure 23.

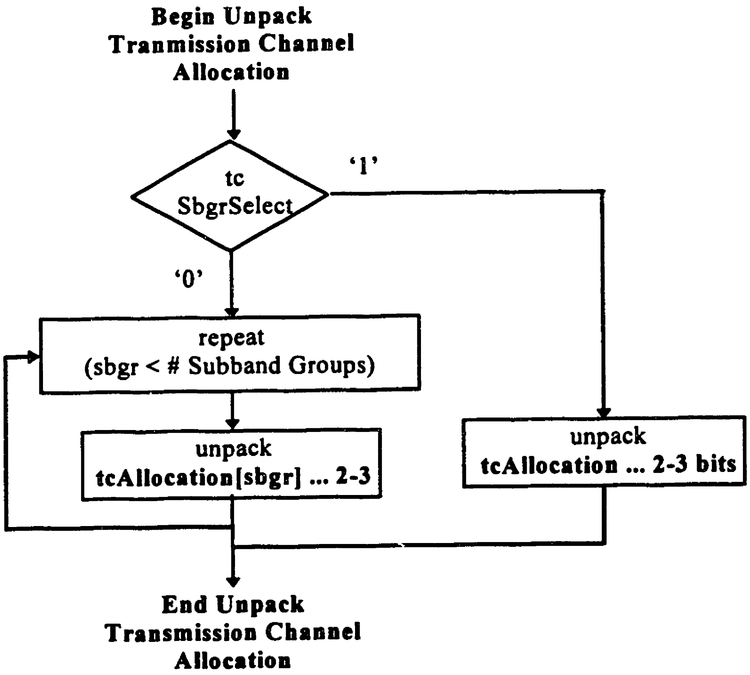
Begin Unpack Tranmission Channel Allocation
tc SbgrSelect
'1'
'0'
repeat (sbgr < # Subband Groups)
unpack tcAllocation[sbgr] ... 2-3
unpack tcAllocation ... 2-3 bits
End Unpack Transmission Channel Allocation

Figure 23 Unpacking Transmission Channel Allocation

The number of bits transmitted for each subband to determine the transmission channel allocation depends on the channel configuration. For the 3/2 configuration shown in Table 12 below, there are eight entries so three bits need to be transmitted for each transmission channel allocation. Fewer bits are required for the other configurations such as the 2/2 configuration (which requires only 2 bits). A complete set of tables for all the configurations is given in the ISO MPEG-2 specification[2]. The MPEG-1 compatible transmission channels, T0 and T1, contain a combination of one of the audio channels not assigned to T2-T4 and two of the audio channels that are assigned to T2-T4. As an example, when the tcAllocation is equal to 0, T0 contains $L^W+C^W+LS^W$ and T1 contains $R^W+C^W+RS^W$. The audio channels are matrixed together for T0 and T1 so that the information contained in the center and surround channels can be heard if the frame is decoded by an MPEG-1 decoder.

3/2 Configuration			
tc Allocation	***T2***	***T3***	***T4***
0	C^W	LS^W	RS^W
1	L^W	LS^W	RS^W
2	R^W	LS^W	RS^W
3	C^W	L^W	RS^W
4	C^W	LS^W	R^W
5	C^W	L^W	R^W
6	R^W	L^W	RS^W
7	L^W	LS^W	R^W

Table 12 Transmission Channel Allocation

Although the audio channels may be assigned to different channels for transmission, it is critical the audio channels consistently be assigned to the same channels for output. This way the left surround channel will always be sent to the left surround speaker of the listener's stereo. The reassignment of the audio channels to the output channels is accomplished as part of the dematrixing function. The transmission channels T0-T4 become these output channels at the end of decoding. The channels are assigned in the manner shown below in Table 13.

Output Channel	*Audio Channel*
T0	L
T1	R
T2	C
T3	LS
T4	RS

Table 13 Output Channel / Audio Channel Correspondence

A table is used to transform the transmitted indexes to a table of transmission channel allocations. Each row of the table, indexed by subband group, identifies which audio channels are assigned to the corresponding transmission channels. The first element in the row is the audio channel contained in T2, the second the audio channel contained in T3, etc. The transmission channel allocation table is used by many of the MPEG-2 functions to efficiently identify the contents of a transmission channel.

3.4.2.2 Dynamic Crosstalk

Dynamic crosstalk is an MPEG-2 procedure which attempts to conserve bits by sharing information between channels. Specifically, multiple channels use the same bit allocation and subband samples but different scalefactors. When dynamic crosstalk is active in a subband group, the requantized but not rescaled samples are copied from one channel to another. The specifics of how samples are reconstructed are discussed in section 3.4.5 on Dynamic Crosstalk. The flowchart below shows the algorithm used to unpack the Dynamic Crosstalk information.

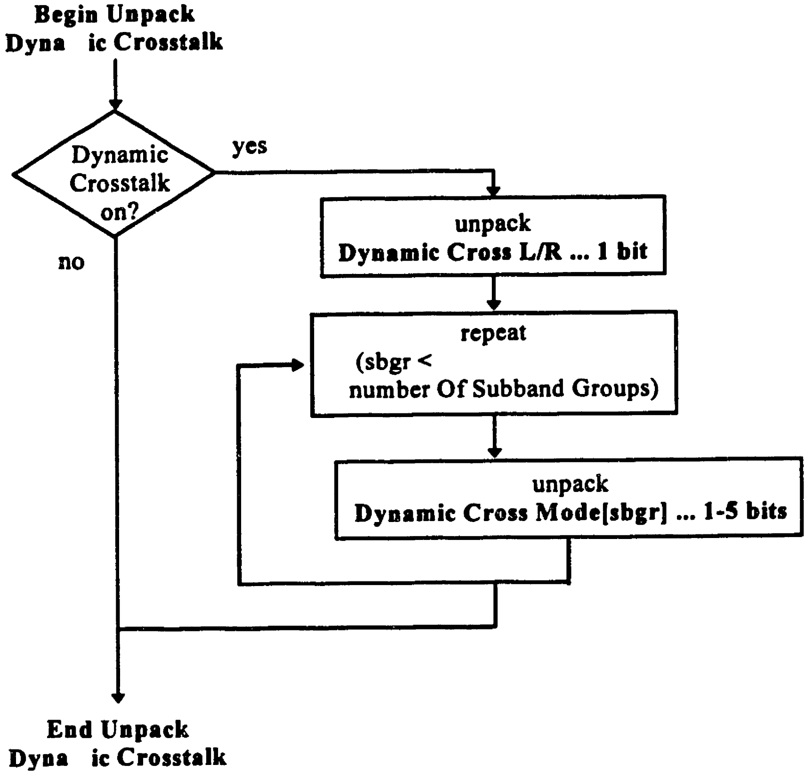

Figure 24 Dynamic Crosstalk Unpacking

The number of bits transmitted for the dynamic crosstalk mode depends on the configuration of the frame; the number can vary from 4 bits for the 3/2 configuration to 1 bit for the 2/1 configuration. A sample dynamic crosstalk table is shown in Table 14 for the 3/1 configuration. The terms T2 and T3 in the table indicate that there is no dynamic crosstalk for those channels (so for the first row there is no dynamic crosstalk). Where a Tij exists in the table, the sample in channel i must be copied to channel j. It is also possible in some tables to have a term Tijk. In this case the sample in transmission channel 'i' is copied to channels 'j' and 'k'. Where there is a '-' in the table, the sample is be copied from either T0 or T1 depending on the contents of the channel. If the channel contains L or LS then the channel is copied from T0; if the channel contains a R or RS then the channel is copied from T1. If the channel contains a center signal, the variable Dynamic Cross Left/Right determines whether the channel is copied from the left side (T1) or the right side (T2).

dynCrossMode[sbgr]	*Transmission Channel*	
0	T2	T3
1	T2	-
2	-	T3
3	-	-
4	T23	-
5	forbidden	
6	forbidden	
7	forbidden	

Table 14 Dynamic Crosstalk 3/1

Configuration

The information unpacked from the bit stream is put into a table so that the dynamic crosstalk channels can be reconstructed efficiently. Each row in the table, indexed by subband group, contains two elements: the number of crosstalk channels that are copied in the subband and an array of structures of type *MCCopyFromTo*. Each *MCCopyFromTo* structure contains the transmission channel that is missing in the *To* element and where to copy the channel from in the *From* element. Since there may be zero through three channels missing in each subband it is important to store the number of missing channels in the *NumberOfCrosstalkChannels* variable. With this table structure, the channels can be efficiently reconstructed. A sample table for the 3/2 configuration is shown below. The number of crosstalk channels may be computed from the appropriate dynamic crosstalk table and it gives the number of From/To structures in the subband group. These are the number of channels that need to be reconstructed in the subband group. The channel that the samples must be copied from is computed from the dynamic crosstalk table and the channel configuration as described in the previous paragraph. Finally the channel that dynamic crosstalk is actually active in may be read directly from the table. So the function reconstructing the samples has the number of copies that need to be performed in each subband and exactly which channels to copy.

Subband Group	number of Crosstalk Channels	From	To	From	To	From	To
0	1	1	4				
1	2	0	3	1	4		
2	1	2	4				
3	0						
4	3	0	2	0	3	1	4
5	0						
6	2	2	3	2	4		
7	0						

Table 15 Dynamic Crosstalk Reconstruction

3.4.2.3 Adaptive Multichannel Prediction

The multichannel prediction section contains part of the information necessary to unpack and decode predicted channels, specifically the variables mcPrediction[sbgr] and predsi[sbgr,px]. These sets of variables are only unpacked if prediction is on in the frame. The array of variables, mcPrediction[sbgr], indicates if prediction is used in each of the specific subband groups. If prediction is on in a specific subband group then several values predsi[sbgr,px], the prediction selects, are also transmitted. The prediction selects indicate how many prediction coefficients are used for each channel in a predicted subband group.

The following diagram shows the variables that are transmitted when prediction is turned on the current frame. When prediction is on, eight mcPrediction[sbgr] variables are transmitted to indicate if prediction is used in each subband group. For each subband using prediction, up to four prediction select values are transmitted.

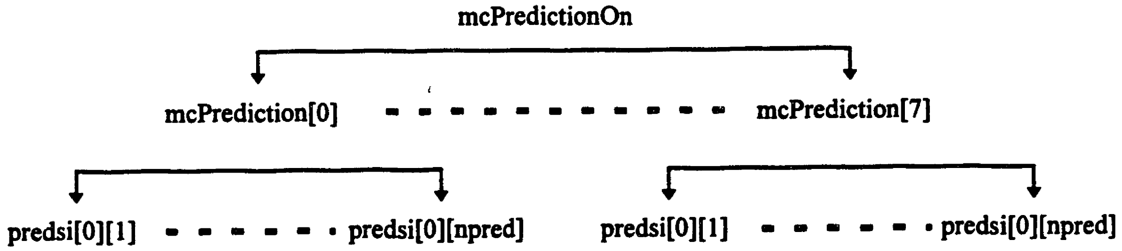

Figure 25 Transmitted Prediction Information

The number of prediction selects transmitted for each subband group depends on the number of transmission channels and the contents of those channels. One prediction select is transmitted for a transmission channel containing a L, R, LS, or RS audio channel; two prediction selects are transmitted for transmission channel containing a center or mono-surround audio signal. The reason for this complication is that the first set of audio channels (L, R, LS, and RS) are predicted from either L0 or R0 while the second set of audio channels (C or S) may be predicted by *both* L0 and R0. The configuration of the system determines the number of transmitted channels while the transmission channel allocation determines the contents of each channel. In addition, if a channel is missing, its subband samples because dynamic crosstalk is active, then prediction may not be used with that channel. A table may be constructed for each configuration, with the transmission channel allocation and dynamic crosstalk as indexes, showing the maximum number of predictors for each subband group. Such a table is shown below for the 3/1 configuration.

tcAllocation	***dynamic crosstalk***							
	0	1	2	3	4	5	6	7
0	4	2	2	0	2	0	0	0
1	3	1	2	0	2	0	0	0
2	3	1	2	0	2	0	0	0
3	3	2	1	0	2	0	0	0
4	3	2	1	0	2	0	0	0

Table 16 Number of Predictors: 3/1 Configuration

In the table above, when the transmission channel allocation and dynamic crosstalk are '0' then channel T2 contains C and T3 contains S. Since both of these channels may be predicted by two different sets of prediction coefficients, four prediction selects are transmitted. When the dynamic crosstalk is changed to '1', the subbands in the channel containing S are no longer transmitted; the channel may no longer be predicted so only two prediction selects are transmitted. If the transmission channel allocation were changed from '0' to '1' while dynamic crosstalk is held at '0', the contents of the transmission channels would then be L and S. L only requires one prediction select so the total number of prediction selects transmitted is three.

Unfortunately this variability in transmitted information results in a decrease of efficiency in unpacking since the unpacking procedure is most efficient if large blocks of data are unpacked at one time. Since it is not known if prediction selects must be unpacked until the variable mcPrediction[sbgr] is unpacked in a subband, the unpacking program size is small and the unpacker must be called many times.

More information about Prediction is contained in section 3.4.3.3 on unpacking prediction from the main audio stream and in section 3.4.6 on how the channels are actually predicted.

3.4.3 Unpacking Multichannel Audio Data

The multichannel audio data section contains the bit allocation, scalefactors, prediction information, and subband samples for up to three multichannels. To unpack the data in these sections, information is needed from the MPEG-1 header (the subband limit), from the Multichannel Header (the number of multichannels), and from the Composite Status section (the status of dynamic crosstalk and prediction).

3.4.3.1 Unpacking Multichannel Bit Allocation

The bit allocations for the multichannels (the audio channels contained in T2-T4) and for the low frequency effects channel are contained in the bit allocation section of the multichannel audio data. The bit allocation for transmission channels T0 and T1, the MPEG-1 compatible channels, were unpacked from the MPEG-1 compatible audio data section.

The number of bit allocations transmitted for the multichannels depend on if the center channel uses phantom coding and the dynamic crosstalk mode for each channel and subband. If the bit allocation for a subband and channel was not transmitted because of phantom center coding or dynamic crosstalk then no bit allocation should be unpacked. For example, no bit allocation is transmitted in a subband of a channel where dynamic crosstalk is active. So no bit allocation should be unpacked for that channel and subband.

The algorithm for unpacking the multichannel bit allocation shown in the flowchart below is used to create an unpacking program. The function begins by checking to see if the frame has a low frequency channel. If it does then a four bit low frequency bit allocation value is unpacked. Then the function begins the main loops for unpacking the bit allocation for each channel and subband. When phantom coding of the center channel is active, no data for subband eleven and higher is transmitted; a transmission channel containing the center channel should not contain a bit allocation for those subbands. A bit allocation is also not transmitted if dynamic crosstalk is active. If neither phantom center channel coding nor dynamic crosstalk are enabled in a certain channel or subband, then the bit allocation is unpacked. The length of the bit allocation for the multichannels depends on the bit allocation table and the subband. The bit allocation table lists the number of bit allocation available in each subband; the number of bits transmitted must increase with the possible number of bit allocations. The existence of the bit allocation also depends on the status of the center channel and on dynamic crosstalk.

The interpretation of the unpacked bit allocation values is the same as in MPEG-1. This process was discussed in the section on unpacking the MPEG-1 bit allocation data

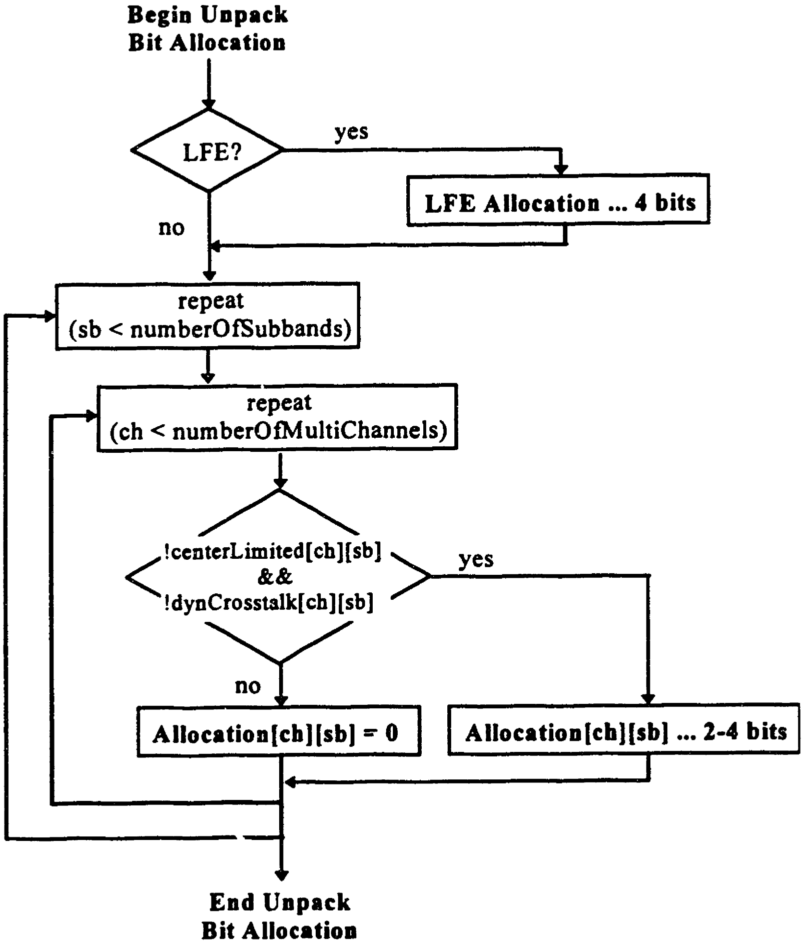

Figure 26 Bit Allocation Unpacking

When all of the transmitted bit allocations have been unpacked, the bit allocation for channels using dynamic crosstalk are reconstructed. The decoder searches through the dynamic crosstalk table in every subband for channels encoded with dynamic crosstalk. For the encoded channels, the bit allocation is copied from the source channel. After unpacking the bit allocation, the channels with dynamic crosstalk will have their proper bit allocation. The bit allocation will be used to determine if the scalefactor selects and scalefactors should be unpacked for each channel and subband.

3.4.3.2 Unpacking Scale Factor Selects

The next section in multichannel audio data contains the scalefactor selects. They indicate the number of scalefactors to be transmitted for each frame. The scalefactor selects for a subband and channel are only unpacked if the subband has a non-zero bit allocation. Scalefactor selects for channels with dynamic crosstalk will be unpacked only if their bit allocations are non-zero. Since the bit allocation is always zero for subbands eleven and higher of a center channel using phantom center coding, no scalefactor selects will ever be unpacked for these subbands. A flowchart of the algorithm to unpack the scalefactor selects is shown in Figure 27, below. To build the unpacking program, the decoder iterates through all the multichannels and subbands and unpacks a scalefactor select for every non-zero quantizer.

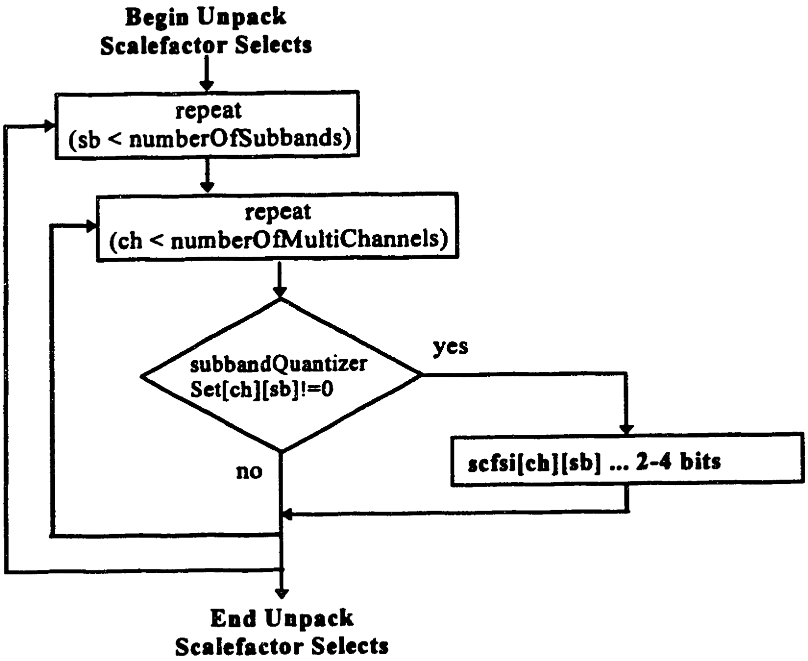

Figure 27 Unpacking Scalefactor Selects

3.4.3.3 Unpacking Prediction Information

Just after the prediction select information in the frame are two data fields containing prediction information; they are the delay compensation values and the prediction coefficients. The three bit delay compensation value specifies a shift of 0,1,2,...,7 samples from the prediction channel. The prediction coefficients are the factors by which the prediction source is scaled to compute the prediction.

A flowchart describing the algorithm for unpacking the prediction information is shown in Figure 28. The prediction information is only included if prediction is active in the frame. The mcPrediction[sbgr] variable is tested to determine if prediction is on in each subband group. If prediction is on in the subband group, then the prediction selects for the subband are tested; if a prediction select is greater than zero, a delay compensation and several prediction coefficients are also transmitted. The number of prediction coefficients transmitted is equal to the value of the prediction select.

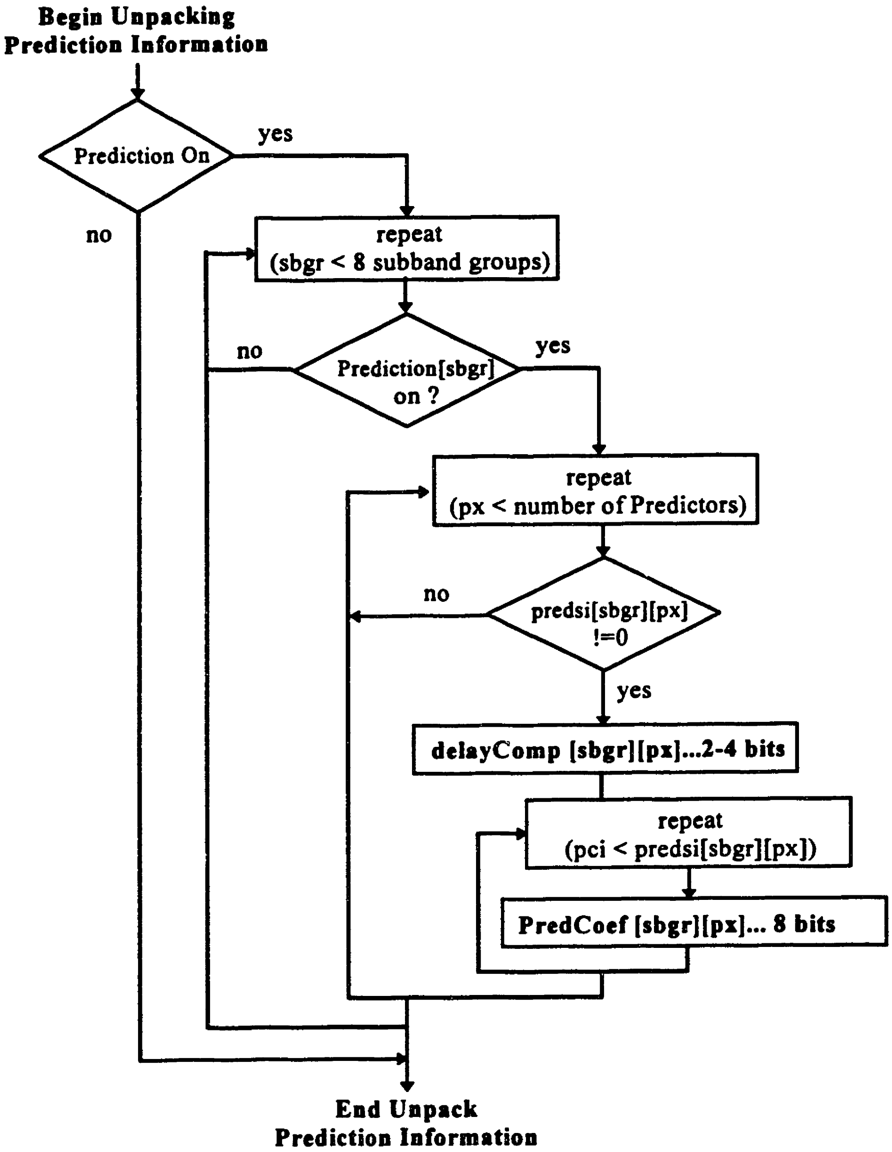

Figure 28 Prediction Unpacking Algorithm

3.4.3.4 Unpacking Multichannel Scalefactors

The next section in the frame contains the scalefactors for the low frequency enhancement channel and the other multichannels. The scalefactor selects previously unpacked are used to decide how to unpack the scalefactors. A flowchart of the algorithm to unpack the scalefactors is shown below in Figure 29.

The algorithm first checks for the presence of a low frequency enhancement channel. If it exists then the appropriate scalefactor is unpacked. Next the scalefactors for all the subbands of the multichannels are unpacked. Since the scalefactors are packed by channel in each subband, the algorithm iterates through the channels inside a loop through the subbands. In each subband and channels, scalefactors are only unpacked if the bit allocation is non-zero. The number of scalefactors (one, two, or three) unpacked for each subband depends on the previously unpacked scalefactor.

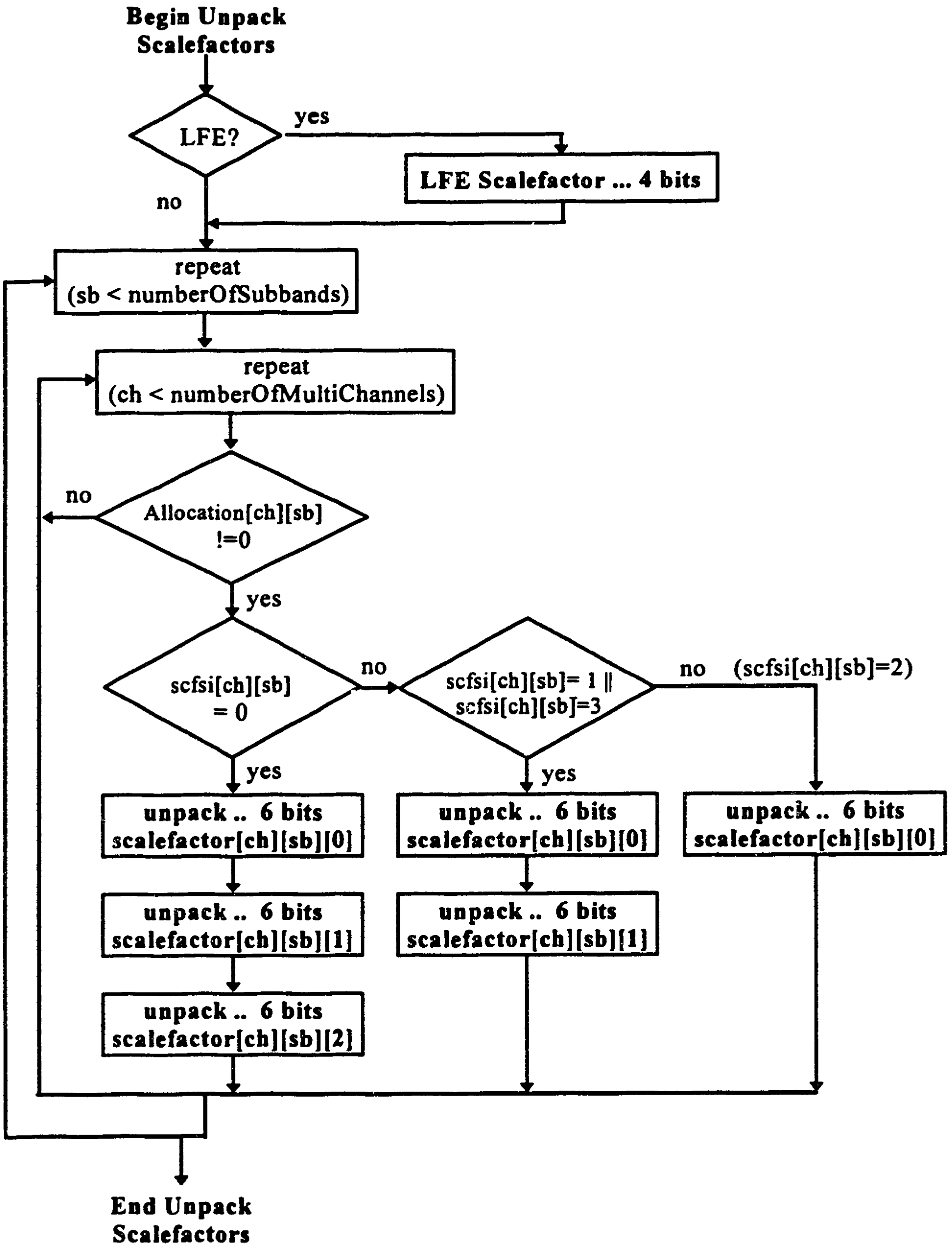
Begin Unpack Scalefactors
LFE?
yes
no
LFE Scalefactor ... 4 bits
repeat (sb < numberOfSubbands)
repeat (ch < numberOfMultiChannels)
no
Allocation[ch][sb] !=0
yes
scfsi[ch][sb] = 0
no
scfsi[ch][sb]= 1 || scfsi[ch][sb]=3
no (scfsi[ch][sb]=2)
yes
yes
unpack .. 6 bits scalefactor[ch][sb][0]
unpack .. 6 bits scalefactor[ch][sb][1]
unpack .. 6 bits scalefactor[ch][sb][2]
unpack .. 6 bits scalefactor[ch][sb][0]
unpack .. 6 bits scalefactor[ch][sb][1]
unpack .. 6 bits scalefactor[ch][sb][0]
End Unpack Scalefactors

Figure 29 Scalefactor Unpacking Algorithm

3.4.3.5 Unpacking Multichannel Subband Samples

The last section multichannel audio data that must by unpacked from the bitstream contains the subband samples. The 36 samples in every subband may be packed in groups of three. Depending on the bit allocation for that subband, the granule of samples may be coded together as a group or placed consecutively in the frame. The decoder must be able to unpack the samples in both cases. The granules for each channel in a subband are packed adjacent to each other.

The algorithm, shown in Figure 30 below, begins by iterating through the 12 granules (36 samples in each frame). On each iteration, one granule of samples must be unpacked from each channel and subband. If the low frequency enhancement is being transmitted, the sample group from it are the first to be unpacked. As the algorithm iterates through each subband and channel, a granule of samples is unpacked if the bit allocation is non-zero. The dynamic crosstalk and phantom center coding cases are covered because the bit allocation for those channels is set to zero and no samples will be unpacked.

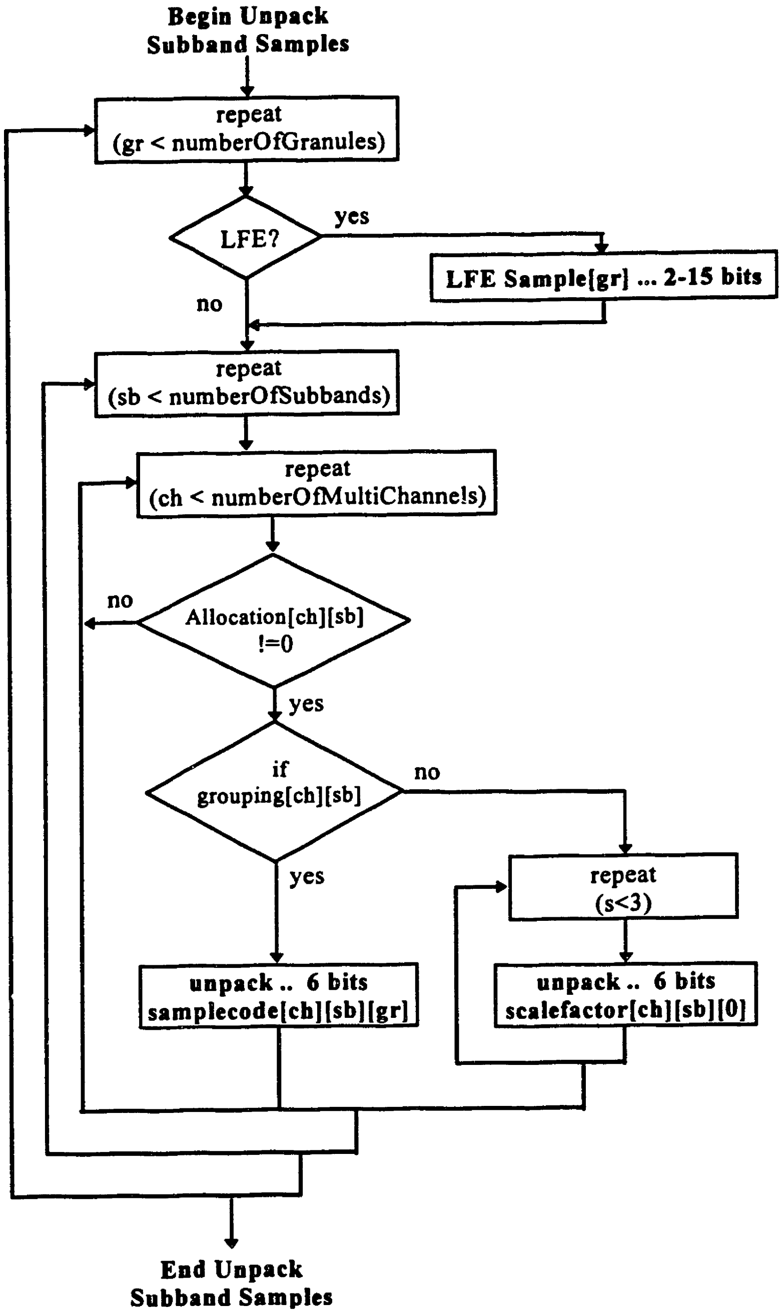

Figure 30 Subband Sample Unpacking Algorithm

3.4.4 Re-synching at End of Frame

At the end of each multichannel frame is a section containing ancillary data. The specific uses this ancillary data section are not defined in the specification. However it is important that they be unpacked so that the decoded can synchronize with the beginning of the next frame. The trick is that there is no variable in the header to define the length of this section. The length of the ancillary data section is difference between the length of the frame (calculated by the samples rate, the bit rate, and the number of samples per frame) the number of bits of data in the frame that have been unpacked up to and including the subband samples. The bits are unpacked as ancillary data and interpreted either as padding bits or as special data unique to a specific encoder/decoder pair.

3.4.5 Dynamic Crosstalk Reconstruction

In the composite status section, the dynamic crosstalk mode for each subband group was transmitted. The modes were also transformed into a table indicating which subbands and channels had dynamic crosstalk active (the destination channel) and from which channel audio data should be copied (the source channel). No bit allocations or subband samples were transmitted for the destination channels, however, bit allocations from the source channel were copied to the destination channels in the unpack bit allocation function. Since all channels with non-zero bit allocations, even channels with dynamic crosstalk active, have scalefactors transmitted, the only task remaining to complete the decoding of dynamic crosstalk is to copy the subband samples. Like the bit allocations, the subband samples from the source channel must be copied to the destination channel. Then the channels with dynamic crosstalk will have information for their samples stored in a format identical to other channels; the dynamic crosstalk channels are decoded exactly like the other channels.

To copy the subband samples, the function must iterate through the dynamic crosstalk table, an example of which was shown in Table 15, by subband and identify which channels are coded with dynamic crosstalk. For the channels coded with dynamic crosstalk, all 36 subband samples must be copied from the source channel to the destination channel. A diagram of the sample structure and the copying of samples is shown below in Figure 31. In this example, dynamic crosstalk is turned on between two channels for subband 0-2 and 8-9. The source channel could be L0 in transmission channel 0 and the destination channel could be LS in transmission channel 2. In the subbands where dynamic crosstalk is active the missing samples are copied from the L0. The original samples exist in the other subbands so no samples need to be copied to them.

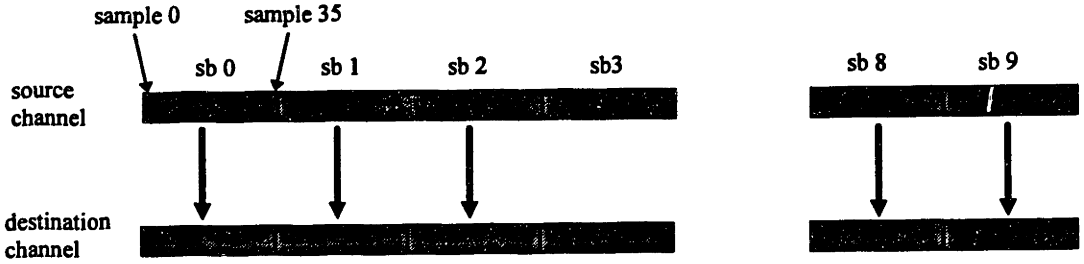

Figure 31 Dynamic Crosstalk Reconstruction

3.4.6 Multichannel Prediction

3.4.6.1 Overview

Any of the multichannels (channels transmitted in T2-T4) may be encoded with prediction. When prediction is used, one of the multichannels is always predicted from one of the MPEG-1 compatible channels. Those channels that are encoded have quantized error signals and prediction coefficients transmitted in the audio data section. This section describes how the encoded channels are reconstructed from the prediction coefficients and the errors.

Subband groups seven and lower of channels T2, T3, and T4 may be predicted from channels T0 and T1 (containing L0 and R0). Exactly which channels are used to calculate the prediction (T0, T1, or both) depends on the contents of the channel being predicted. Center (C) or mono Surround (S) audio channels may be predicted from both T0 and T1 while the Left (L), Left Surround (LS), Right (R), and Right Surround (RS) may be predicted from either T0 or T1. In fact the audio channels on the left side (L and LS) are always predicted from T0 and the audio channels on the right side (R and RS) are always predicted from T1.

The method for determining where prediction is active and how many predictors are used was discussed in section 3.4.2.3. For a subband and channel where prediction is turned on, an error signal is transmitted instead of subband samples. The error value is the difference between the calculated prediction and the actual sample values. The error values are necessary since unless the predicted channel is perfectly correlated with the first, the prediction coefficients cannot be chosen to exactly predict the channel.

3.4.6.2 Computation

There are two steps to computing the predicted channels: first the prediction is calculated by taking the product of the prediction coefficients and samples of the source channel and second the error signal transmitted in the predicted channel is added to the prediction. If there is more than one prediction coefficient then the prediction for each sample from each prediction coefficient is summed. The samples in the source channel and the error signal should both be requantized and rescaled before carrying out the prediction.

The equations for the prediction of the Center channel and the Right Surround channel are shown below. There is one term in each summation for every prediction coefficient. In the calculation of the center channel prediction there are two summations since it is predicted from both T0 and T1. The equation for the prediction of the Right Surround channel only has one summation since it is predicted from only T1. There is a delay in the source sample term if the delay Compensation value is not equal to zero or if the number of predictors is greater than one.

$$\begin{aligned} C^{\lambda}[n] = \sum_{k=0}^{2} predCoefC0[sbgr][k] * T0[n - delayComp - k] + \\ \sum_{k=0}^{2} predCoefC1[sbgr][k] * T1[n - delayComp - k] \end{aligned} \tag{26}$$

$$RS^{\lambda}[n] = \sum_{k=0}^{2} predCoefRs[sbgr][k] * T1[n - delayComp - k] \qquad (27)$$

The samples in the channel are equal to the predicted values plus the error transmitted in the channel.

$$C[n]=C^{\lambda}[n]+\varepsilon_{c}[n] \qquad (28)$$

$$RS[n]=RS^{\lambda}[n]+\varepsilon_{rs}[n] \qquad (29)$$

3.4.6.3 Example

In the 3/2 configuration, T2, T3 and T4 contain C, LS, and RS respectively. The first two of the four transmitted prediction selects apply to T2 while the second two apply to T3 and T4. A delay value and a group of prediction coefficients is transmitted for each non-zero prediction select (the number of prediction coefficients is equal to the value of the prediction select). If the four prediction selects transmitted are '1', '1','0', and '2', then samples in channels T2 and T4 are predicted while the actual samples are transmitted for channel T3. Two delay compensation values and two prediction coefficient are associated with channel T2, the first delay compensation and prediction coefficient for prediction from channel T0 and the second for predicting from T1. One delay compensation value and two prediction coefficients are associated with T4. To calculate the predicted values of the center channel, the decoder uses the L0 and R0 samples and the first two prediction coefficients. To calculate the predicted values of the RS channel, the decoder sums the product of samples from R0 with the two associated prediction coefficients. The final C and RS signals are calculated by adding the errors transmitted in the T2 and T4 channels to the predictions.

3.4.6.4 Delays Greater than Zero

The current decoder and all known encoders use only zero order prediction. Any delay greater than zero requires samples from the end of the previous frame to carry out the prediction. To efficiently integrate the stored samples into the current streamlined procedure seems difficult at best.

3.4.7 Dematrixing

The audio channels are combined together in the encoder to produce MPEG-1 compatible channels which contain information from the multichannels as well as the Left and Right channels. The channels must be dematrixed in the decoder. In addition to removing the multichannel information from T0 and T1, the dematrixing function also reverses transmission channel allocation by placing audio channels in their proper transmission channel (L is placed in T0, C is placed in T2, LS is placed in T4, and so forth).

Since the dematrixing function reverses the transmission channel allocation, the function depends on the transmission channel allocation as well as the transmitted matrix procedure index. If fact, when the transmission channel allocation changes, so must the dematrixing function. Two example dematrixing equations for dematrixing procedure zero for the 3/2 configuration are shown in the table below. The equations in the table below still subtract out the same channels as the equations given in section 2.6. The equations below differ only because they subtract out transmission channels instead of audio channels.

tcAllocation	*Decoding Matrix*		*tcAllocation*	*Decoding Matrix*
0	L^{W}=L0-T2-T3		1	C^{W}=L0-T2-T3
	R^{W}=R0-T2-T4			R^{W}=R0-C^{W}-T4
	C^{w}=T2			L^{w}=T2
	LS^{W}=T3			LS^{W}=T3
	RS^{W}=T4			RS^{W}=T4

Table 17 Dematrixing Equations '0' - 3/2 Configuration

When the tcAllocation is equal to 1, the Left channel is assigned to T2 and the Center channel is derived from T0. The dematrixing function must calculate C^{W} and store the result in T2, while moving L to T0, without overwriting these channels. To accomplish this, one of the channels must be stored to a temporary register. With different assignments to the temporary register, all of the dematrixing procedures may be decoded efficiently.

The first step in dematrixing is to determine the dematrixing procedure to be used. In the current version of the TI decoder, matrix procedures 0, 1, and 3 have been implemented. A table associated with each dematrixing procedure contains the information to dematrix each subband based on the transmission channel allocation. The rows of the dematrixing table are indexed by the transmission allocation; each row contains the source channel of the temp register, the destination channel of the temp register, the three channels to dematrix L0, the channel in which the place the result of dematrixing L0, the three channels to dematrix R0, and the channel in which to place the result of dematrixing R0. Thus the structure contains all of the data necessary to dematrix the channels. By changing the channel assigned to the Temp register, the calculation of the result register, and the replacement of both of these registers, this functional block can be used to carry out all of the dematrix procedures.

An example of the function for dematrixing the channels (in the case where the tcAllocation is equal to one) is shown below. First the Temp register is filled with the contents of the temp source pointer. The Temp source is a channel that may be overwritten by the result of the dematrixing. When the tcAllocation='1' for the subband, T2 is channel assigned to the Temp register, since the dematrixed Center channel will be written into T2. The next step in the algorithm is to carry out the dematrixing of channel L0. In this case, T2 and T3 are subtracted from T0. The result is stored in the Result register. Next, the Temp register is stored in the proper destination; in this case the contents of the Temp register (the left channel) is moved to T0. Last, the Result register is stored in T2- so the Center channel is placed in the correct transmission channel. After these four steps, L0 has been dematrixed and the associated audio channel placed in the correct transmission channels. In the description above, the MPEG-1 compatible left channel was dematrixed, a similar procedure must carried out to dematrix the right channel.

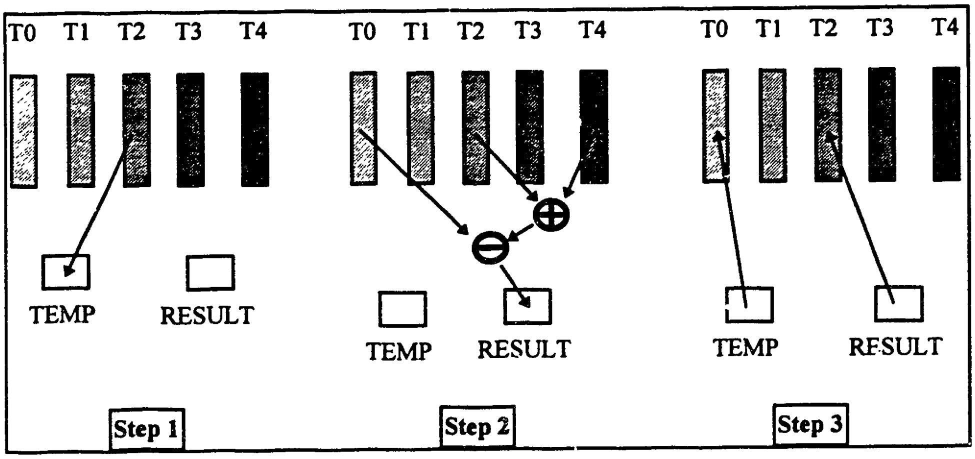

Figure 32 Dematrixing Function

3.4.8 Denormalizing

When the multichannels are matrixed together to create the two MPEG-1 compatible channels in the encoder, the center and surround signals may be attenuated by several different factors. In matrix procedures zero and one, the multichannels are divided by √2; in matrix procedure one the surround channels are divided by 2 and the center channel is divided by √2. The weighting is the only difference between matrix procedure zero and one; the surround channels will be emphasized slightly less in procedure one.

Since the MPEG-1 compatible channels could contain a higher amplitude signal after matrixing, they are normalized so that the maximum amplitude of the combined signal remains the same. for example, in matrix procedure '0' the maximum signal would occur when L, C, and LS have unit magnitude and are in phase. Then the amplitude of the combined signal would be:

$$\begin{aligned} L0 &= L + \frac{1}{\sqrt{2}}C + \frac{1}{\sqrt{2}}LS \\ &= 1 + \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} \\ &= (2+\sqrt{2})\sqrt{2} \end{aligned} \tag{30}$$

If the channel is divided by the denormalizing factor, 1+√2, then the maximum amplitude remains one.

The table below shows the unweighting and denormalizing factors for all the dematrixing procedures. To denormalize the samples, the decoder iterates through the subbands and channels, and multiplies each sample by the unweighting factor and the denormalizing factor.

dematrix Procedure	*signals*	*unweighting factor*	*denormalizing factor*
0,2	L^W,R^W	1	1+√2
	C^W,LS^W,RS^W	√2	
1	L^W,R^W	1	
	LS^W, RS^W	2	1.5 + .5*√2
	C^W	√2	
3	L^W,R^W,C^W,LS^W,RS^W	1	1

Table 18 Unweighting and Denormalizing Factors

3.4.9 Extension Bit Stream Implementation

The capability to decode extension bit streams requires several procedures to maintain extension stream parameters and changes to the bit unpacker function to allow unpacking from multiple streams. The remainder of this section discusses the functions to maintain the extension bit stream parameters while section 3.4.11 deals with the changes to the unpacking function.

3.4.9.1 Overview

The unpacking of an MPEG-2 frame which uses an extension bit stream includes several new steps. When multichannel header for a frame is decoded and an extension stream is to by used, a function to initialize the extension stream parameters is called. The function makes sure that the extension stream is synchronized, unpacks the extension header, and calculates the length of the extension frame. Using the length of the extension stream, words are read from the extension bit stream into a buffer. Pointers are assigned to designate the current word in the buffer, the last word of valid data in the buffer and the end of each frame in the buffer. Then the decoder returns to unpacking the multichannel data from the MPEG-1 compatible frame. The unpacking proceeds as in a frame without an extension bit stream until the end of the MPEG-1 compatible frame is reached. Then the unpacker function must automatically switch to unpacking from the extension stream. A flowchart of the extension stream unpacking process is shown below.

If an error is detected in the extension header or in the data of the extension stream, the extension stream is re-synchronized by stopping the unpacking of the current frame and searching for the synchronization word at the beginning of the next frame. If there was no error in the MPEG-1 compatible part of the stream or in the extension stream, the output may be displayed.

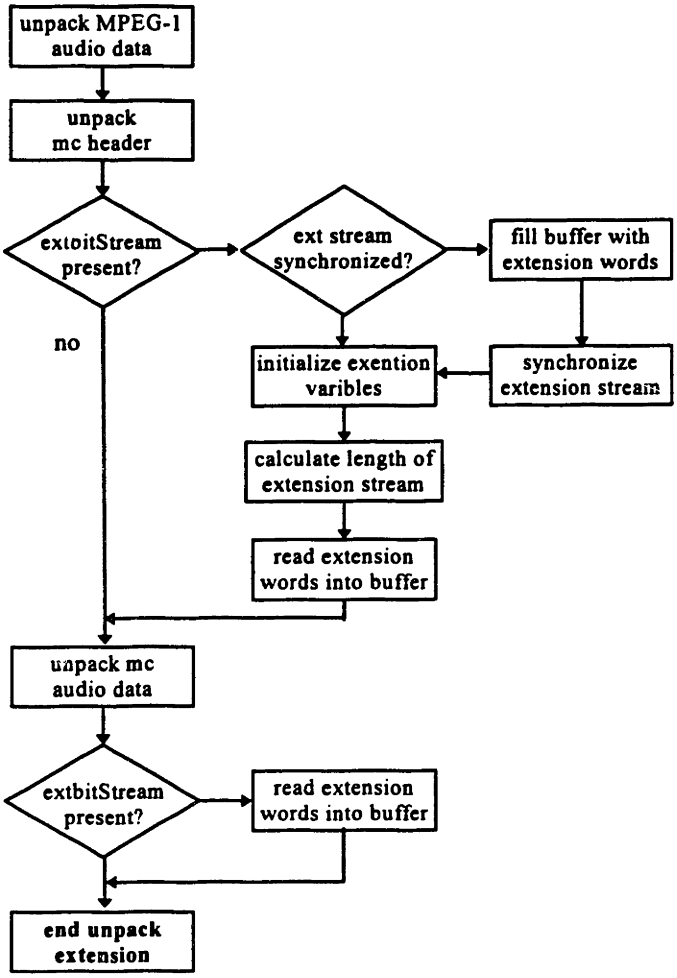
unpack MPEG-1 audio data
unpack mc header
extoitStream present?
ext stream synchronized?
fill buffer with extension words
no
initialize exention varibles
synchronize extension stream
calculate length of extension stream
read extension words into buffer
unapack mc audio data
extbitStream present?
read extension words into buffer
end unpack extension

Figure 33 Extension Bit Stream Flowchart

3.4.9.2 Extension Variable Initialization

If an MPEG frame uses an extension bit stream, the first step is to check to see if the extension frame has been synchronized. That is, if a synchword has been found in a previous frame and the frame is aligned so that the extension header is contained in the first element of the buffer. The extension stream will never be synchronized on the first frame of an MPEG stream and could become unsynchronized if an error occurs. If the stream is not synchronized a function is called which fills the extension buffer and scans for an extension synchword. Once the synchword is found and the frame is synchronized, variables must be set up which describe the extension word buffer.

The extension word buffer contains the words that have been read in from the extension bit stream. When the words are unpacked it is necessary to have a pointer to indicate to the unpacking function the next word to be unpacked. This pointer is called the buffer pointer. Another pointer, the end of buffer pointer, is used to denote the end of the frame. The end of buffer pointer is used to calculate the number of words remaining in the frame and to make sure that no data is unpacked past the end of the buffer. A third pointer is used to indicate the last word of data in the frame. This pointer is used to make sure that no data is unpacked past the end of the current frame. Since it is important to be able to unpack data exactly to the end of the frame to keep the frame synchronized, and since the frame does not necessarily end on a word boundary, a variable used to store the number of bits of data in the last word that belong to the current frame. The ancillary data for the frame is unpacked exactly up to this point.

3.4.9.3 Extension Processing Procedure

The extension stream processing procedure is responsible for unpacking the extension header, reading words into the extension buffer to unpack the current frame, and calculating the exact frame boundary of the main data stream and the extension stream.

The items to be unpacked in the extension stream header have been previously described in overview of the extension stream. One element of the extension header, the Extension Length is used to compute the number of data words that need to be read into the buffer. The number of words to read into the buffer is the difference between number of words required for decoding the next frame and the number of words already in the buffer. The number of words required is given by:

```
numberOfWordsRequired= (extensionLength*8 bits
                              - number of bits remaining in current word            (31)
                              + 32 bits) /
                              (number of bits in a word);
```

To calculate the number of words that must be read in the buffer, the difference is taken between the number of words already in the buffer and the number of words required. When the new words have been read into the buffer, the pointer to the end of the buffer and the pointer to the end of frame are updated. The exact end of the frame is calculated so that the data may unpacked to the end of the frame with single bit accuracy.

3.4.9.4 Unpacking Ancillary Bits, Synchronizing For Next Frame

After the multichannel subband samples have been unpacked from the extension data stream the remaining bits in the frame are defined as ancillary data bits and are also unpacked. The use of the ancillary data bits is undefined, so they are treated as padding bits between the end of data and the end of the frame; they are unpacked and discarded. The number of padding bits is the number of bits originally in the extension frame minus the number of bits unpacked so far from the extension frame. After the padding bits are unpacked, the next bits to be unpacked are the frame header for the next frame. If there was an error in the calculation of the number of padding bits, then the next frame will not be properly aligned and will need to be re-synchronized. If less than the actual number of padding bits were unpacked, then the data pointer will be in front of the next frame, and the frames will be re-synchronized properly. If more bits were unpacked, then the data pointer will be past the next header and re-synchronization will cause a frame to be skipped. In a full MPEG system, the problem would be solved by checking the presentation time stamps of each frame.

3.4.10 MPEG-2 CRC Check Implementation

It is important to compute the CRC for MPEG-2 since that is the only way to determine if a valid multichannel frame exists. In decoding a frame an MPEG-2 decoder always assumes that the frame contains multichannel data. The multichannel header information, the composite status, bit allocation, and scalefactor selects must all be unpacked and used to compute the CRC word. If the computed word matches the word transmitted in the stream, then valid multichannel data does in fact exist and decoding continues. If the CRC check fails, then either an error occurred in the transmission of the data or no multichannel data was transmitted. The MPEG-1 compatible data for the current frame is decoded and pointers to the end of the MPEG-1 compatible data are restored. If the stream is only MPEG-1 the decoder should be aligned for the next frame; if the stream contained multichannel data with an error, the decoder will search through the multichannel data for the next frame synchword.

3.4.10.1 Protected Bits

The protected bits in MPEG-2 include the bits in the multichannel header, the composite status section, the bit allocation, and the scalefactor selects. It is impossible to compute the number of bits used in the calculation before unpacking through the scalefactor selects since the number of bits describing the transmission channel allocation, dynamic crosstalk, and prediction are variable. Thus the decoder has to unpack and process several sections of data before it can determine if it is decoding a valid multichannel frame.

3.4.10.2 Error Handling

If the calculated CRC value does not match the transmitted CRC value for one the CRC checks in MPEG-2 an error for than section is signaled. If the error occurred at the beginning of the multichannel data then the unpacking of the multichannel data should stop and the subband samples for the multichannels set to zero. Since no error occurred in the MPEG-1 compatible channels, processing should take place normally with those channels and they should be played. An error in the MPEG-2 Extension stream is treated in a similar manner; the multichannels should be muted but the MPEG-1 compatible channels may be played.

3.4.11 MPEG-2 Unpacker Function

While the previous section 3.4.9 described the functions required to maintain the extension bit stream parameters, this section describes the changes that must be made to the unpacking function to be able to unpack data from multiple streams.

3.4.11.1 Unpacking From Multiple Streams

The unpack function must be modified for decoding frames using extension bit streams to decode from more than one input stream. It also possible to encode the MPEG-1 compatible part of a frame with layer I coding and the multichannel portions of the frame with layer II (the multichannel portion may only be encoded with layer II). Since the multichannel portion will be three times as long, one complete multichannel frame is transmitted for every three complete MPEG-1 compatible frames. In decoding, the four data sections may be treated as independent data streams. Thus the unpack function has been designed to unpack data from up to four bit streams.

3.4.11.2 Bit Accuracy Switching Between Streams

A frame does not need to end on a word boundary, and the bits being unpacked for a data field may also be split between streams. Therefore a modification must be made to the structure describing the input word buffer so the last bit in the last word of the stream may be specified. The unpacker must be modified so that it switches to the next stream when the last bit is reached.

3.4.11.3 Unpacking Efficiency

When the unpack function is unpacking any of the large fields of data, bit allocation, subband samples, etc., most of the unpacker cycles are spent in the loops where bits are being unpacked from the current word. It is desired that no cycles be added to this critical section of code. When the unpacker reaches a word boundary, it checks to see if the next word is the end of the frame. If it is, then the unpacker switches to a second loop in the function which facilitates a switch between streams. Since this second loop is only executed near the end of the frame the number of cycles spent in the second is insignificant compared to the number of cycles spent in the first loop. All of the additional operations necessary for unpacking from multiple streams are added in the second unpacking loop.

3.4.11.4 Maintain Variables for multiple streams

It is important to maintain the variables for all of the streams. When the unpacker switches streams, it is important that the shift, bit, and buffer variables in that stream are correct so that unpacking for the next frame will begin in the proper place. For example when unpacking a frame using an extension bit stream, at the end of the MPEG-1 compatible frame, the unpacker must switch to the extension stream. After the remainder of the frame has been unpacked from the extension stream, the decoder must switch back to the main data stream to start unpacking the next frame. The decoder must store the exact location unpacking stopped in the MPEG-1 compatible stream so that the frames remain synchronized.

4. Experimental Results - UNIX Implementation

Two types of experiments were carried out to verify that the UNIX version of the MPEG-2 audio decoder was working properly: perceptual listening tests and comparisons with the output of other decoders. The other decoders used in the comparisons were a limited functionality ISO MPEG-2 decoder and a TI MPEG-1 decoder. Since the ISO decoder was not able to decode most bit streams, it was difficult to directly verify that the TI decoder was correctly decoding streams using an external bit stream, dynamic crosstalk, or multichannel prediction. These functions were only verified by perceptual listening tests. Plots of the bit streams using matrixing and dynamic crosstalk are shown to illustrate the effect of these functions on the channel signals.

Informal listening tests were conducted in two ways: first on a multimedia PC with a Soundblaster card and a set of headphones and second with a surround sound system connected to a workstation. These tests were most useful when first testing the decoder to make sure there were no dramatic errors. To conduct the tests, a decoded channel from each of the decoders was loaded into the audio player. The two channels were played alternately to see if there were any differences. Few errors were actually discovered this way, but it was a useful technique to see that the decoding was basically correct. It was also an important technique since even though the exact values of the outputs may be slightly different; the perception of the outputs should be exactly the same.

The differences between the two decoders were tested more precisely by comparing the outputs with the MATLAB Numeric Computation and Visualization Software. In MATLAB, two tests were used to compare the outputs: a bit difference measurement and a signal-to-noise measurement. In the bit difference measurement, the two signals were aligned and subtracted. The result shows where the two signals were different. The signal-to-noise measurement determines the perceptual significance of the difference between the two signals. The signal-to-noise ratio is small when the ratio of the error variance to the signal variance is large. If there is no difference between the two signals, the error would be zero and the signal-to-noise ratio would be infinite. The signal to noise ratio is

$$SNR = 10\log_{10}(\frac{\sigma_x^2}{\sigma_e^2}) \qquad (32)$$

where σ_x^2 is the variance of one of the signals and σ_e^2 is the variance of the difference between the two signals. The variance of the signals is computed over 384 samples, the length of a layer I frame or one third the length of a layer II frame.

4.1 Direct Decoder Comparisons

4.1.1 TI/ISO Comparison of Base Bit Stream

A base MPEG-2 stream (a stream not using an extension bit stream, dynamic crosstalk or multichannel prediction) was decoded with the TI UNIX decoder and the ISO UNIX decoder. The ISO decoder is an MPEG-2 audio decoder developed by IRT in Munich, distributed to members of the MPEG committee. The ISO decoder only decodes base MPEG-2, layer II streams at the 48 kHz sampling rate and 384 Kbits/s bit rates and will not decode streams using an extension bit stream, dynamic crosstalk, or multichannel prediction. The ISO code is also not designed to run in real time on any processor.

The base bit stream is an excerpt of a Mahler symphony recorded live at Radio France, encoded and distributed by CCETT. A copyright notice appears in the Appendix. Shown below in Figure 34 is the center channel output of the TI decoder, the center channel output of the ISO decoder and the difference between the two. There are many instances where there are differences between the two samples, but the magnitude of the difference is no greater than one. These differences may be accounted for by the use of different levels of accuracy in the bit allocation and scalefactors, and in different methods of converting floating point numbers to integers (The TI version truncates numbers while the ISO code rounds).

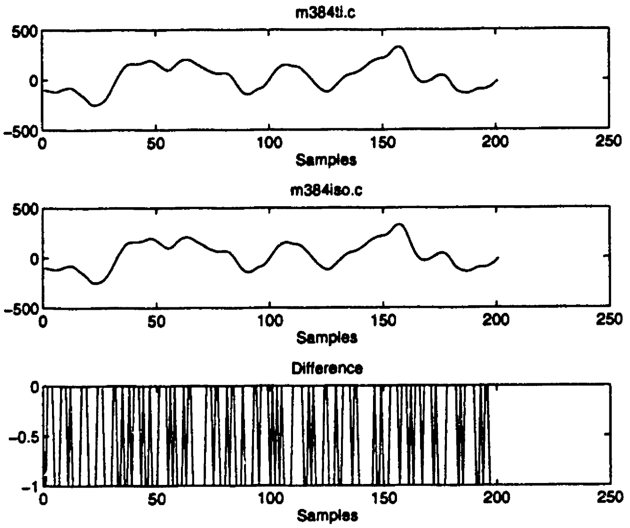

Figure 34 Center Channel Difference

The left channel output of the TI and ISO decoders is shown in Figure 35 below. The difference between them is used to calculate the signal-to-noise ratio. Although there are many differences between the two outputs, the signal-to-noise ratio is around 150 dB. The difference between the signals from the two decoders can be thought of as an additive noise signal. Since the noise signal is 150 dB below the signal power, the noise should also be well below the masking threshold. Thus the two signals should be perceptually identical to the listener.

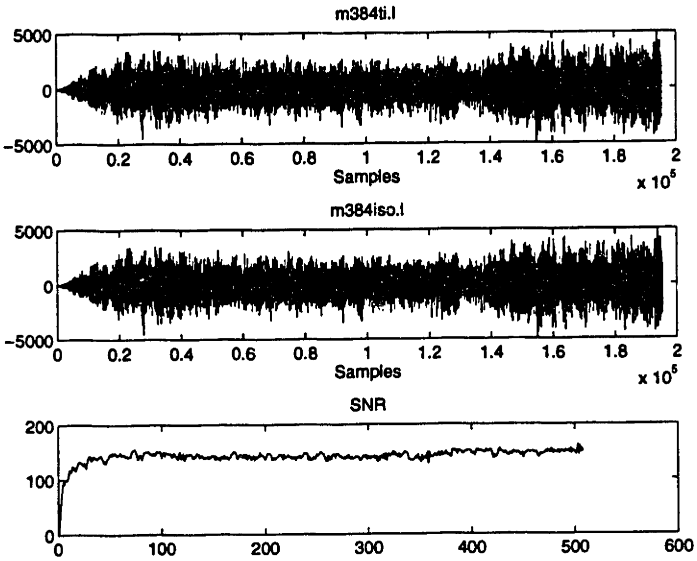

Figure 35 Left Channel SNR

4.1.2 TI MPEG-2 / TI MPEG-1 Comparison

The other direct test of the decoder is to compare the outputs of the TI MPEG-2 decoder with the TI MPEG-2 decoder for an MPEG-1 bit stream. This demonstrates the forwards compatibility of the MPEG-2 system: an MPEG-2 decoder should be able to decode a mono or stereo bit stream encoded with an MPEG-1 encoder.

The beginning of every frame of MPEG data has a synchword, however, the beginning of the MPEG-2 part of the frame has no synchword. The only way for an MPEG-2 decoder to correctly decode an MPEG-1 frame is to assume the frame is a MPEG-2 frame, decode all of the multichannel information, perform a CRC check on the multichannel data, and decide that the frame is only MPEG-1 if the CRC check fails. The decoder must also be sure to read what it assumed was multichannel data as MPEG-1 data for the next frame. The output of a portion of the decoded file is shown in Figure 36 below. There are no differences between the outputs from the two decoders.

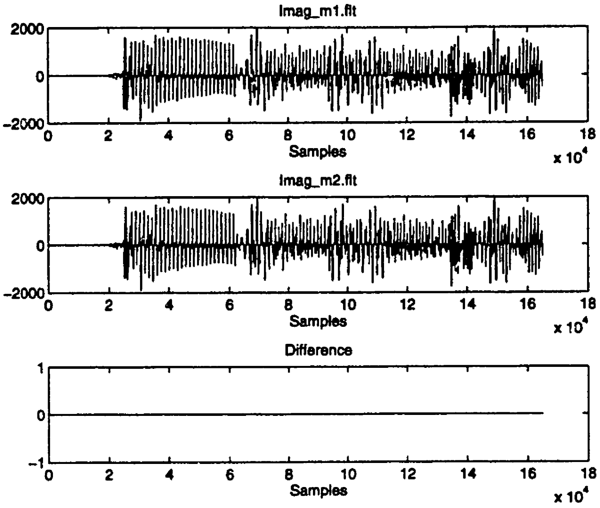

Figure 36 MPEG-1 File

4.2 MPEG-2 Functions

4.2.1 Matrixing

The dematrixing function was complicated to code and to debug. Fortunately the ISO decoder supported three of the four dematrixing functions so they were all verified to work correctly.

In the m384 stream, the transmission channel allocation is '0' for about 80% of the frames so the channel allocation and the dematrixing equations are:

Transmission Channel	T0	T1	T2	T3	T4
Audio Channel	L	R	C	LS	RS

Table 19 Transmission Channel Allocation '0'

Decoding Matrix
L=L0-T2-T3
R=R0-T2-T4
C=T2
LS=T3
RS=T4

Table 20 Dematrixing Equations

The multichannels are not affected by dematrixing for transmission channel allocation '0'. However in 20% of the frames, the transmission channel allocation is '3'. The transmission channel allocation changes so the left surround channel is transmitted in T0 and the left channel is transmitted in T3. The left surround channel is calculated in the dematrixing calculation: LS=L0-T3-T2 and the result is stored in T3. The contents of T3, the left channel, are moved to T0. Figure 37 shows part of the action of dematrixing, The top graph shows the left surround channel before dematrixing, the middle graph shows the left surround channel after dematrixing, and the bottom graph shows the difference between the first two. The samples that are different are from frames where the transmission channel allocation has been changed from '0' to '3'.

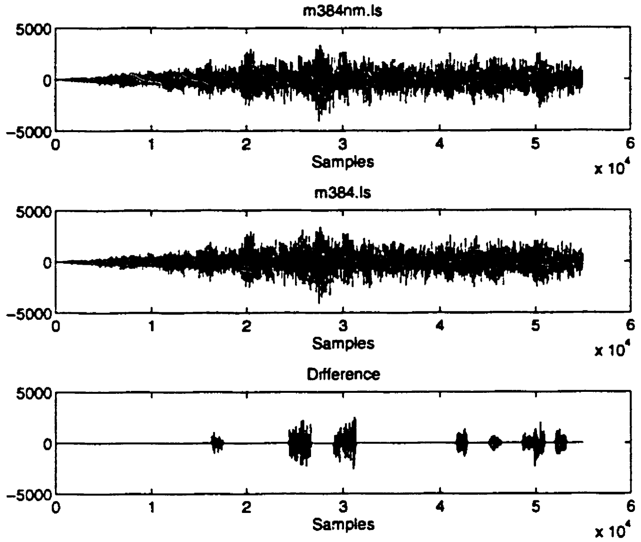

Figure 37 Dematrixing

4.2.2 Prediction

Since the prediction function was not i plemented in the ISO decoder, there was no way of directly testing to determine if the channel prediction was correct. Testing was done perceptually by listening to the output and examining the bit stream and the samples before and after prediction. The stream titled 'pred384.mpg', encoded by IRT in Munich, was a short clip of several people singing. All three of the multichannels are predicted from the L0 and R0 channels.

When the stream was played without decoding the prediction, all of the channels contained many audible artifacts. Even though only the multichannels can be predicted, an error in decoding the prediction will show up in the other channels after dematrixing (the matrixing was carried out in the encoder before prediction so results of the dematrixing operations in the decoder will contain errors if channel prediction was carried out incorrectly). When the prediction decoding function was implemented, the number of audible artifacts was significantly reduced. After debugging, the remainderwere assumed to be

Figure 38 below shows the prediction error, the resulting signal, and the prediction. The objective in prediction is to minimize the error, so the error signal may be quantized with fewer bits than the original signal. The coding gain for each subband is proportional to the difference in energy between the signal and the error in that subband. The total coding gain for all of the subbands is

$$G_M = \frac{\left(\prod_{k=0}^{M-1} \sigma_{xk}^2\right)^{1/M}}{\left(\prod_{k=0}^{M-1} \sigma_{ek}^2\right)^{1/M}} \tag{33}$$

where M is the number of subbands, σ_e^2 is the variance of the error, and σ_x^2 is the variance of the signal[15].

The prediction errors are in general smaller than the original signal, but in three locations there are large spikes. These spikes are locations where the prediction algorithm broke down in the encoder, that is where there were significant differences between the signal and the prediction. When the channel is played after prediction, there are still audible clicks at these locations.

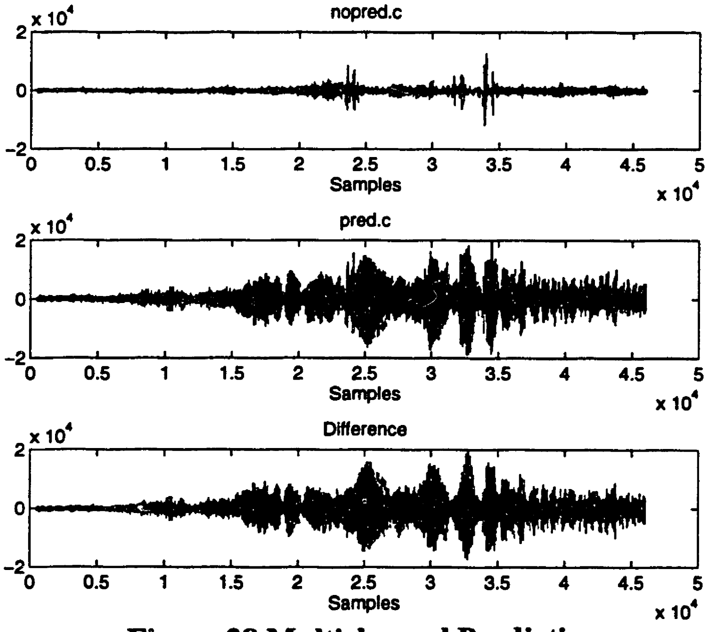

Figure 38 Multichannel Prediction

4.2.3 Dynamic Crosstalk

Since dynamic crosstalk decoding has not yet been implemented in the ISO version of the decoder, the implementation in the TI decoder was tested with perceptual listening tests and by examining the bit stream. The source of data for the tests was a stream encoded with dynamic crosstalk, m512dypc.mpg, from CCETT in Paris. Again making verification difficult was the fact that the original bit stream was also not available for comparison.

However, by examining the bit stream, it can be determined what dynamic crosstalk operations must be carried out, which subband sample and bit allocations are missing and which must be copied, and if the correct operations are being performed. Thus by manually examining the bit stream, the implementation of dynamic crosstalk was verified.

The output of the decoder was also played on a surround sound stereo system. Although there were some background artifacts, no sharp clicks or other indications of a problem with dynamic crosstalk were heard.

4.3 Conclusions

Testing the multichannel functions was a laborious process involving a lot of time examining bit streams. However the process verified the implementation of dematrixing, multichannel prediction, dynamic crosstalk, and compatibility with MPEG-1. Many more interesting tests need to be performed examining the effectiveness of the multichannel functions. For instance it would be interesting to test how much dynamic crosstalk improves the quality at a given bit rate or if multichannel prediction introduces unacceptable artifacts. These tests were not able to be performed due to a lack the original source material and a fully functional MPEG-2 encoder.

5. Experimental Results - C30 Implementation

In addition to the UNIX implementation, the decoder was implemented on a TI Digital Signal Processing chip. Two types of experiments were conducted with the C30 version of the decoder: experiments to verify that the decoder was working properly and experiments to benchmark the efficiency of the decoder. The definition of working properly was producing an output stream that sounds good to a listener and was statistically similar to output of the TI UNIX decoder. Section 5.2 presents the results of the verification. The efficiency of the decoder was determined by measuring the number of cycles required by each function to decode a frame of data, a procedure known as benchmarking. Benchmarking was used to determine which sections of the program used the most instruction cycles; the code in these sections was examined and if possible improved. Benchmarking was also used to determine if the code would run in real time, this aspect is discussed in section 5.3. The number of cycles needed to decode a frame also gives an indication of the complexity and the cost of implementing the algorithm in silicon, an important consideration at Texas Instruments.

5.1 DSP Implementation

The decoder was implemented on a Texas Instruments TMS320C30 DSP. No new code was written for the DSP implementation; the UNIX code was compiled to produce DSP assembly code. Since the UNIX and C30 architectures and compilers were significantly different, the C30 version was extensively tested to verify that it was working properly. The C30 implementation was benchmarked to evaluate its performance, these results are described in a following section. Following the initial benchmarking, hand coded assembly routines for the subband filtering operations were compiled into the decoder to improve the performance.

5.2 Verification of the DSP Implementation

The C30 version of the decoder was verified to produce a proper output in two ways: by conducting perceptual listening tests and by comparing the output of the C30 decoder with the output of the TI UNIX decoder in MATLAB. The perceptual tests were less precise, but since it was an audio decoder it was important to listen to the output. The comparison in MATLAB was useful to quantify where and how many errors occurred. The tests were applied to a stream encoded with the standard MPEG-2 encoder (no extra coding techniques), a stream encoded with multichannel prediction, and a stream encoded with dynamic crosstalk and an extension bit stream. The tests were run comparing the C30 decoder with the TI UNIX decoder, but the results would have been similar with the ISO decoder.

5.2.1 Basic Stream: m384.mpg

The file m384.mpg is a 5 channel stream encoded with only the standard MPEG-2 functions (no extension bit stream, dynamic crosstalk or prediction). After the stream was decoded, a perceptual listening test, a bit difference test and a SNR test were all performed.

The result of the perceptual listening test was that the difference between the two output streams was imperceptible. Both streams contained artifacts like screeches and clicks, but no additional artifacts seemed to be introduced by decoding on the C30.

The MATLAB test was more precise and more revealing. Both signals were loaded into the program; the original signals and the difference are shown in the figure below. The difference between the two signals ranges between plus and minus one with many differences occurring at the beginning of the signal and only sporadic differences in later frames of the signal. The errors at the beginning of the signal are due to differences in the computation of the synthesis window when the buffers are mainly empty. The errors in later frames are likely due to differences in precision and rounding between the two systems. For example, to improve the efficiency of the C30, it truncates when converting floating point numbers to integers.

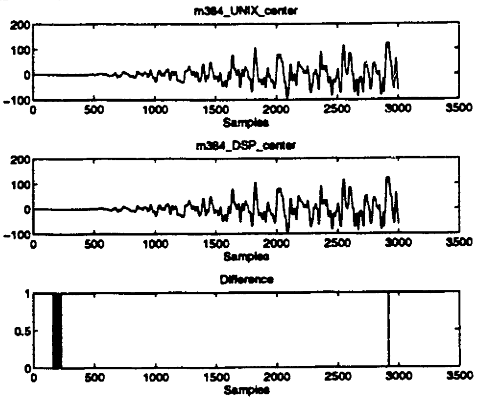

Figure 39 Center Channel Comparison: m384

In the first frame where the amplitude of the signal is low, between +/- 5, several errors occurred. Even though the amplitude of the errors was also small, the signal-to-noise ratio was rather poor. When averaged over the first 384 samples the signal-to-noise ratio was computed to be 17 dB. Since neither the amplitude nor the frequency of the errors increase with the amplitude in later parts of the signal, the SNR is constant at around 200 dB for the rest of the frame. In fact, many of the frames are identical so the SNR may not even be computed since the noise signal is zero.

5.2.2 Extension Bit Stream and Dynamic Crosstalk: m512dypc

Since the file m384.mpg was encoded with only the minimum set of MPEG-2 functions, it is important to test another stream which uses the extension bit stream, dynamic crosstalk and prediction functions. The file m512dypc.mpg uses both the extension bit stream and dynamic crosstalk. The left channel, decoded by the UNIX and C30 versions of the decoder is shown below.

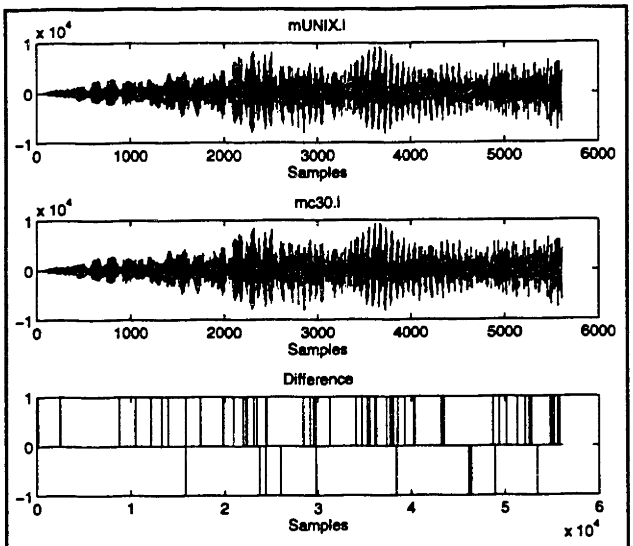

Figure 40 Extension Bitstream with Dynamic Crosstalk

We would expect more differences between the two streams when more calculations take place on the data. By examining the bit stream, it is seen that the dynamic cross modes are all '8', '1', or '0'. From the dynamic crosstalk tables, it is found that the only channel encoded with dynamic crosstalk will be the T4, the right surround channel. The RS channel should have more bit differences since more processing is done on that channel by the decoder. Indeed the SNR for the right surround channel ranges between 180 dB and 200 dB, significantly less than the other channels. The differences are still not significant enough to be audible.

5.2.3 Multichannel Prediction: pred384.mpg

The file pred384.mpg is a 40 frame long MPEG-2 stream which is encoded using prediction. Like the other two files, pred384.mpg was decoded with both the UNIX and C30 version of the decoder. The output of each was compared perceptually and with MATLAB.

The listening test showed that there were no large differences in the two streams. The encoding of the stream was poor- there was a lot of static and some artifacts that sounded like popping. No additional artifacts were apparent in the C30 version.

The streams were next loaded into MATLAB. One item of interest was that differences at the beginning of the frame were gone. The stream had a long section of silence at the beginning where the decoders only produced a zero output. Since the data is zero at the beginning of the frame in this stream, the section where there is normally a difference, is muted out. The figures also show that there are differences of plus or minus one spread throughout the rest of the stream. The signal-to-noise ratio for the two streams ranges between 150 and 240 dB.

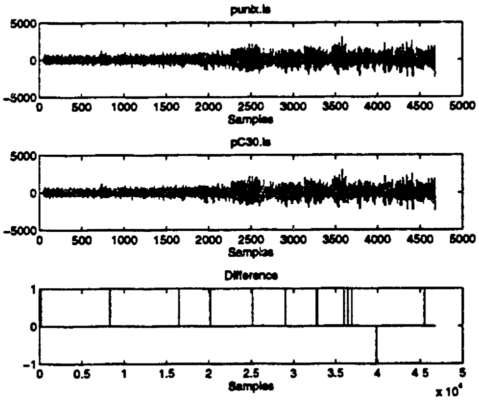

Figure 41 Multichannel Prediction

5.3 Benchmarking Results

The second set of tests performed on the C30 version of the compiler, called benchmarking tests, determined the number of cycles used by each procedure. The objective is to examine the number of cycles used when each function is called to tell which sections of code are most heavily used. For heavily used sections, either the C code may be modified to compile into more efficient code, or the C code may be re-written in assembly. The number of cycles to decode a frame determines if the decoder is efficient enough to run in real time on a processor.

Benchmarking tests were first conducted on a bit stream with two different versions of the C30 decoder; one version had replaced some heavily used sections of C code with assembly code. The tests show the improved efficiency of the decoder with assembly code. Next bit streams using dynamic crosstalk, multichannel prediction, extension bit streams were decoded to determine the number of cycles required to decode these MPEG-2 Functions.

The code will run real time if the number of cycles to decode a frame does not exceed the number of cycles available per frame on the DSP chip. The number of cycles available for decoding each frame if the length of the frame in time seconds multiplied by the number of operations per second (there are two clock cycles for every operation)

$$\frac{cycles}{frame} = \frac{\sec onds}{frame} * \frac{clockcycles}{\sec ond} * \frac{1}{2} \tag{34}$$

A layer II frame is .024 s long, so there are 396,000 cycles per frame with a 33 MHz part or 720,000 cycles per frame on a 60 MHz part. One additional consideration is that the number given by benchmarking does not include I/O functions to read data in and out of the decoder. The I/O functions should take about 70,000 cycles; so the goal for making the decoder run in real time should be about 650,000 cycles per frame.

5.3.1 Standard Bit Stream

To analyze the output of the benchmarking reports, functions are grouped together by use. The group labeled Unpacking MPEG-1 contains the functions that unpack the frame header, the protection word, the bit allocation, the scalefactors, and the subband samples. Unpacking MPEG-2 contains the functions for unpacking the multichannel header, bit allocation, scale factors, prediction information, and subband samples. The MPEG-1 functions are dequantizing, rescaling, and denormalizing the subband samples; these are function that operate on all channels of data but were defined in the MPEG-1 specification. The MPEG-2 functions are dynamic crosstalk, multichannel prediction, dematrixing, and denormalizing. The last group of functions carry out the subband synthesis operations.

In the first test, a bit stream with only the basic MPEG-2 functions (no dynamic crosstalk, multichannel prediction, or extension stream) was decoded. The version of the decoder contained no assembly routines. The number of cycles used by the functions in the frame are given below in Table 21. Since subband synthesis filter uses 82% of cycles for the frame; the filter should by optimized.

Function	*Number of Cycles*
Unpacking MPEG-1	65435
Unpacking MPEG-2	85535
MPEG-1 Functions	98227
MPEG-2 Functions	104898
Synthesis Filtering	1616117
Total Decoding	1970212

Table 21 Decoding Cycles

5.3.2 Standard assembly procedures

One way to increase the speed of execution is to re-write critical sections of C code directly in C30 assembly language. Since writing in C30 is slow and complicated, it is best to re-write only the most heavily used sections of code. For the MPEG-1 compiler, the sections of code that implement the synthesis window take up a large percentage of the cycles. A section of code written in TMS320C30 assembly language by Jon Rowlands of Texas Instruments, Inc. was substituted for the C code. The same basic bit stream was decoded using the assembly procedures. A graph of the cycles used by the different functions is shown in Figure 42 below. The assembly versions of the synthesis window require only 350 thousand cycles while the 'C' versions require 1.6 million instructions. Synthesis filtering has been reduced to 50% of the total cycles for the frame and the total number of cycles required to decode the base MPEG-2 frame are reduced from nearly 1.9 million to 699 thousand cycles.

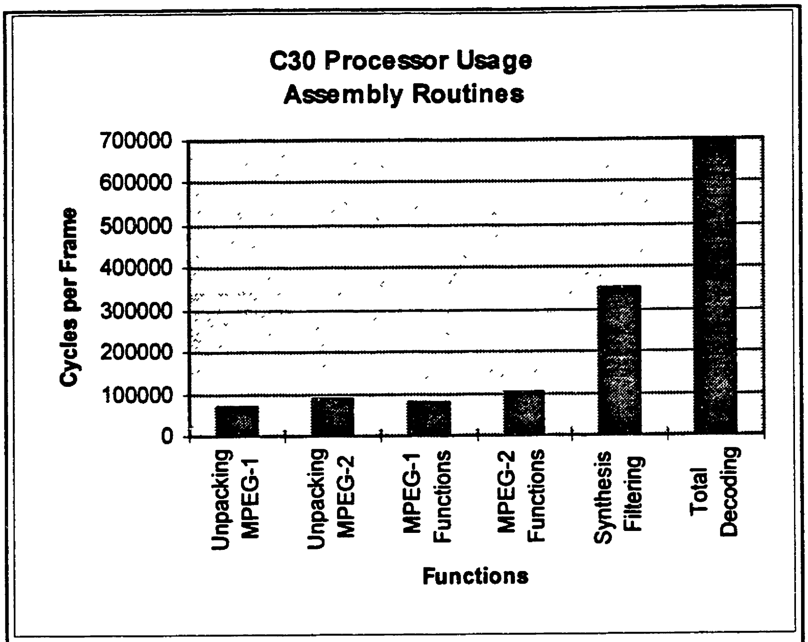

Figure 42 Decoder Cycles with Assembly routines

5.3.3 Extension Bit Stream

Several changes occur when an MPEG file using an extension bit stream is decoded. First, a section of code must be executed to setup and maintain the second bit stream. Second, since there are more bits available to encode the data, the number of cycles required to decoded the multichannel data and the MPEG-1 compatible data increases. A file with an extension bit stream would contain the same number of bit allocations, but more of the bit allocations would be set to non-zero values and more samples would be unpacked. As shown in the table below, the number of cycles required to unpack the bit allocations is roughly constant between the two types of files, but the number of cycles to unpack the subband sample raises dramatically in the file using the extension bit stream.

Function	*No Extension Stream*	*Extension Stream*
unpackBitAllocation	8457	8425
unpackSubbandSamples	40118	49560
unpackMPEG2BitAllocation	12954	12865
unpackMPEG2SubbandSamples	44766	46938

Table 22 Extension Bit Stream Decoding Cycles

It is also interesting to note that the number of cycles required to process the samples, the cycles for the categories MPEG-1 and MPEG-2 functions, have not increased at all. Although the resolution of the quantization of the samples has increased, the number of samples has not. An additional 13,000 cycles are used to process the extension bit stream; that is to maintain the extension buffer and carry out the other functions described in the section on the extension bit stream implementation. The figure of 13,000 does not included the cycles required for reading the extension file. It is also interesting to note that the number of cycles in the most cycle consuming function, the synthesis filter, remains the same. While the over 25,000 additional cycles are only three percent of the total decoder time, the new cycles are critical since the decoder is already close to the real time cycle limit.

Function	*No Extension Stream*	*Extension Stream*
Unpacking MPEG-1	68586	77838
Unpacking MPEG-2	88770	91832
MPEG-1 Functions	80107	79932
MPEG-2 Functions	104898	104898
Ext Bit Stream		13376
Synthesis Filtering	351274	351274
Total Decoding	693635	719150

Table 23 Extension Bit Stream Decoding Cycles

5.3.4 Dynamic Crosstalk

The purpose of this section is to compare the differences in cycles between an extension file and an extension file encoded with dynamic crosstalk. As described in the dynamic crosstalk description, for channels where dynamic crosstalk is active, scalefactors are transmitted but the bit allocations and samples are not (they are copied from another channel). The bits saved by sharing information are used to more accurately code the transmitted samples both in the MPEG-1 compatible channels and in the multichannels. As shown in the table below, the cycles for unpacking the all of the MPEG-1 compatible data increased only slightly. No large differences are expected since the MPEG-1 compatible channels may not be encoded with dynamic crosstalk. The next section shows the cycles for unpacking the multichannel data. Neither the bit allocations nor the samples are transmitted for the channels encoded with crosstalk. As expected, the cycles to unpack the bit allocation drops, but the cycles to unpack the subband samples actually increases. Due to the larger number of bits available, there are probably more non-zero bit allocations and consequently more samples than need to be unpacked. The function to decode the crosstalk channels is efficient, and only 2788 instructions.

Function	***Crosstalk Off***	***Crosstalk On***
unpack itAllocation: B	8425	8523
unpackScaleFactors:	16220	17085
unpackSubbandSamples:	49560	50675
unpackMPEG2BitAllocation:	12865	10522
unpackMPEG2ScaleFactorSelects:	11171	10758
unpackMPEG2ScaleFactors:	14160	14534
unpackMPEG2SubbandSamples:	46938	48488
undoDynamicCrosstalk:	159	2788
Total Decoder Time	724777	731570

Table 24 Dynamic Crosstalk Decoding Cycles

5.3.5 Prediction

The file pred384.mpg, coded using multichannel prediction, was benchmarked to determine the number of cycles used by functions to decode multichannel prediction. A comparison of the number of cycles used in different functions was made with the file m384.mpg, which was coded without using prediction (or the extension bit stream). The results of the comparison are shown in the table below. The *UnpackMPEG-2CompositeStatus* function unpacks the values that determine in which subbands prediction is active and the prediction selects which determine how many prediction coefficients are used for each channel in the subband. The *UnpackPrediction* function unpacks the delay Compensation values, the delay between the predicted channel and the source channel, and the prediction coefficients. Most of the cycles are consumed by the *PredictChannels* function which uses the values unpacked in the previous two functions to predict the missing channels. The total number of cycles used to carry out the prediction is almost 50,000 with 27% of the cycles used to unpack the prediction data 72% of the cycles used to calculate the predicted samples.

Function	*Without Prediction*	*With Prediction*	*Difference*
UnpackMPEG2CompositeStatus	3062	11371	8309
UnpackPrediction		5202	5202
PredictChannels		36329	36329
Total	3062	52902	49840

Table 25 Prediction Decoding Cycles

5.4 Conclusions

The perceptual listening tests and the MATLAB comparisons of the data show that the C30 version of the compiler produces nearly the some output as the UNIX version. The small differences of +/-1 at the beginning of the frame are inaudible. The decoder is close to running in real time when decoding a base MPEG-2 file. The software may break the real-time barrier if several critical functions, such as the bit stream unpacker, are rewritten in TMS320C30 assembly language.

6. CONCLUSIONS

An MPEG-2 audio decoder implementation was completed in C on a UNIX workstation and on a TMS320C30 Digital Signal Processing chip. The decoder implemented all of the MPEG-2 functions in at least a basic form.

In this implementation, the decoder was able to decode streams encoded with a MPEG-1 audio encoder, demonstrating the backwards compatibility function. The decoder was able to decode streams with a dynamic transmission channels allocation and those streams using matrixing procedures zero, one and three. The fourth matrixing procedure, compatible with Dolby Surround Sound encoding should be implemented in a future version of the decoder. The decoder supports first order multichannel prediction with no delay compensation. This level of prediction was as extensive as any of the encoders available for testing. Dynamic crosstalk decoding was fully implemented. Finally, extension stream decoding was fully supported and the decoder was able to decode streams with a combined bitrate of up to 896 Kbits/s.

The C30 version of the decoder was benchmarked to evaluate the performance of the individual functions and to find the total number of cycles to decode each frame of audio. The majority of cycles for decoding each frame were consumed in the subband filtering operations. Using hand-written assembly code for the filtering function, the total number of cycles to decode each frame of audio was reduced to 699,000 cycles. This figure does not include input/output operations which may consume another 70,000 cycles. For a 60 Mhz DSP chip, the total number of cycles available to decode each layer II, 48 kHz frame is 720,000. Thus the decoder does not quite run in real time on a 60 Mhz DSP. To get the decoder to run in real time, either a faster DSP is required, or more of the functions must be re-written in assembly code.

The testing of the decoder was limited by the lack of a fully functional MPEG-2 encoder and good bit streams. For instance, the bit stream available for testing prediction only used first order prediction with no delay compensation. In addition, the quality of the prediction stream was very poor. The best way to advance encoder and decoder technology would be to advance them as a pair. Then each new function be thoroughly tested. Also a fully functional encoder would make perceptual testing much more useful. Since original bit steams were not available (i.e. the same input audio stream coded with and without prediction) it was difficult to determine the perceptual effects of the different multichannel coding techniques. To work on these problems, a functional encoder must be developed.

In the MPEG-2 audio system, of the encoder/decoder pair, it is expected that there will initially be a larger market for the decoder. In an application like satellite broadcasting of television and associated audio for home theater, one high quality encoder may produce a signal which is broadcast to hundreds of thousands of decoders. So initially sales of the decoder will probably be the main source of revenue, but it will still be necessary to have a functional encoder to fully develop and test the decoder. As multichannel audio recording becomes more popular in camcorders or in video conferencing, the market for the complete encoder/decoder system will also grow.

7. APPENDIX

```
/*****************************************************/
/*                                                    */
/*      MPEG2 AUDIO MULTI-CHANNEL BIT STREAMS         */
/*                                                    */
/*      Layer II - 48 kHz - 3/2 presentation          */
/*                                                    */
/*      Freeware - Copyright (c) CCETT 1994           */
/*                                                    */
/*      Departement TAV/SVS - jbrault@ccett.fr        */
/*                                                    */
/*****************************************************/
```

Several of the bit streams used in testing the decoder were encoded by CCETT in Paris. The above copyright notice appears per their request. The bitstreams used in testing consisted of four excepts of a Mahler symphony recorded live at Radio France (Paris).

The bit streams are:

M384.MPG

layer	: II
total bitrate	: 384 Kbits/s
sampling frequency	: 48 kHz
number of subbands	: 27

MPEG1 COMPATIBLE PART

bit rate	: 384 Kbits/s
mode	: stereo

MULTICHANNEL EXTENSION

center channel	: present
surround channels	: Stereo
LFE channel	: none
presentation matrix	: -3dB surround -3dB center
multilingual channels	: 0
additional bitstream	: none
dynamic_crosstalk	: off
prediction	: off

M384pc.mpg

layer	: II
total bitrate	: 384 Kbits/s
sampling frequency	: 48 kHz
number of subbands	: 27

MPEG1 COMPATIBLE PART

bit rate	: 384 Kbits/s
mode	: stereo

MULTICHANNEL EXTENSION

center channel	:	with phantom coding
surround channels	:	Stereo
LFE channel	:	none
presentation matrix	:	-3dB surround -3dB center
multilingual channels	:	0
additional bitstream	:	none
dynamic_crosstalk	:	off
prediction	:	off

m512.mpg m512.ext

layer	:	II
total bitrate	:	512 Kbits/s
sampling frequency	:	48 kHz
number of subbands	:	27

MPEG1 COMPATIBLE PART

bit rate	:	384 Kbits/s
mode	:	stereo

MULTICHANNEL EXTENSION

center channel	:	present
surround channels	:	Stereo
LFE channel	:	none
presentation matrix	:	-3dB surround -3dB center
multilingual channels	:	0
additional bitstream	:	yes
extension stream length	:	384 bytes
MPEG1 ancillary length	:	3 bytes
dynamic_crosstalk	:	off
prediction	:	off

m512dypc.mpg m512dypc.ext

layer	:	II
total bitrate	:	512 Kbits/s
sampling frequency	:	48 kHz
number of subbands	:	27

MPEG1 COMPATIBLE PART

bit rate	:	384 Kbits/s
mode	:	stereo

MULTICHANNEL EXTENSION

center channel	:	with phantom coding
surround channels	:	Stereo
LFE channel	:	none
presentation matrix	:	-3dB surround -3dB center
multilingual channels	:	0
additional bitstream	:	yes
extension stream length	:	384 bytes

MPEG1 ancillary length	:	3 bytes
dynamic_crosstalk	:	on
prediction	:	off

References

1. ISO/IEC 11172-3, "Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at Up To About 1.5 Mbits/s: Audio", *International Organization for Standardization*, 1993.

2. ISO/IEC 13818-3, "Information Technology - Coding of Moving Pictures and Associated Audio: Audio", *International Organization for Standardization*, 1994.

3. J. Rowlands, "Digital Audio Compression", *1994 Custom Integrated Circuits Conference*, 1994.

4. D. Esteban and C. Garland, "Application of Quadrature Mirror Filters to Split Band Voice Coding Schemes," *ICASSP '77*, pp. 191-199, 1977.

5. J. Rothweiler, "Polyphase Quadrature Filters - A New Subband Coding Technique", *ICASSP '83*, Boston, pp. 1280-1283.

6. J. Princen, A. Johnson, A. Bradley, "Subband/Transform Coding Using Filter Bank Designs Based on time Domain Aliasing Cancellation", *ICASSP 1987*, pp. 2161-2164.

7. K. Brandenburg, "Perceptual Audio Coding", AES Convention 1992, preprint 3336.

8. J. Rowlands. private conversation May 10, 1995.

9. A. V. Oppenheim, Course Lecture: Cambridge, April, 1994.

10. A. V. Oppenheim and R. W. Schafer, Digital Signal Processing. New Jersey: Prentice Hall, 1989. pp117-122.

11. MPEG-1 Spec., pp. 28, pp. B2-B5.

12. MPEG-1 Spec, p. 40.

13. H.S. Hou, "A fast recursive algorithm for computing the discrete cosine transform", *IEEE Trans. Acoust., Speech, and Signal Proc.*, Vol. ASSP-35, No. 10, pp 1455-1461, October 1987

14. K. Konstantinides, "Fast subband filtering in MPEG audio coding", *IEEE Signal Proc. Lett.*, Vol. 1, No. 2, pp. 26-28, February 1994

15. P. Vaidyanathan, *Multirate Systems and Filter Banks*. New Jersey: Prentice Hall, 1993. pp. 830-833.

THESIS PROCESSING SLIP

FIXED FIELD ill ______ name ______

index ______ biblio ______

► COPIES [✓ Archives] Aero Dewey [✓ Eng] Hum

Lindgren Music Rotch Science

TITLE VARIES ►□ ______

NAME VARIES: ►☒ Henry

IMPRINT (COPYRIGHT) ______

►COLLATION: 121 p

►ADD. DEGREE: ______ ► DEPT.: ______

SUPERVISORS: ______

NOTES:

cat'r: date:

page:
►J104

►DEPT: E.E.

►YEAR: 1995 ►DEGREE: M.Eng

►NAME: MIKKELSON, Chad